Journal of Artificial Intelligence Research 74 (2022) 393-458 Submitted 09/2021; published 05/2022

Objective Bayesian Nets for Integrating Consistent Datasets

Jürgen Landes JUERGEN_LANDES@YAHOO.DE
Munich Centre for Mathematical Philosophy
Open Science Center
Ludwig-Maximilians-Universität
Munich, Germany

Jon Williamson J.WILLIAMSON@KENT.AC.UK
Department of Philosophy and Centre for Reasoning
University of Kent
Canterbury, United Kingdom

Abstract

This paper addresses a data integration problem: given several mutually consistent datasets each of which measures a subset of the variables of interest, how can one construct a probabilistic model that fits the data and gives reasonable answers to questions which are under-determined by the data? Here we show how to obtain a Bayesian network model which represents the unique probability function that agrees with the probability distributions measured by the datasets and otherwise has maximum entropy. We provide a general algorithm, OBN-cDS, which offers substantial efficiency savings over the standard brute-force approach to determining the maximum entropy probability function. Furthermore, we develop modifications to the general algorithm which enable further efficiency savings but which are only applicable in particular situations. We show that there are circumstances in which one can obtain the model (i) directly from the data; (ii) by solving algebraic problems; and (iii) by solving relatively simple independent optimisation problems.

1. Introduction

It is increasingly common to collect multiple datasets, involving hundreds of variables and thousands of observations, to address a single problem. Different datasets tend to measure different variables, even when the datasets are collected with the same application in mind. For instance, it is common in systems pharmacology—and indeed in systems medicine more generally—to have datasets measuring proteomics, transcriptomics, metabolomics, clinical data, and patient-reported outcomes, and for these datasets to have very few variables in common; see, e.g., Bai and Abernethy (2013); De Pretis et al. (2021); Tricco et al. (2016). How do we integrate all this data?

One approach to data integration is motivated by Objective Bayesian Epistemology (OBE), which holds that a rational agent ought to adopt as a representation of her degrees of belief the probability function with maximum entropy, $P^\dagger$, from all those probability functions that fit her evidence. The entropy of a probability function is a measure of the extent to which it equivocates between possible outcomes, and this approach is usually justified on the grounds that $P^\dagger$ is the function that fits the evidence but is maximally non-committal or equivocal in other respects (Jaynes, 2003; Williamson, 2010; Landes and Williamson, 2016).

In this paper, we apply OBE to the situation in which the agent's body of evidence consists of a collection of datasets (and nothing else). We take all variables to be discrete and we assume that the datasets have been gathered in such a way that, when a variable occurs in more than one dataset, it

is genuinely the same variable, with the same number of values, measured in the same way, in each dataset in which it occurs. Furthermore, we assume that the datasets are large and reliable enough that each dataset distribution provides an accurate estimate of the frequency distribution of the measured variables, and that they are consistent in the sense that these marginal frequency distributions are satisfiable by some joint probability function defined on the set V of all the variables measured by the datasets.[1]

The agent's belief function $P^\dagger$ will be defined on the algebra generated by this larger set V of variables. OBE holds that $P^\dagger$ should agree with each marginal distribution of measured frequencies, and should otherwise have maximum entropy. (To the extent that the dataset distributions only approximate the corresponding marginal frequency distributions, the maximum entropy function $P^\dagger$ which agrees with the dataset distributions should be thought of as an approximation to the belief function warranted by OBE (Williamson, 2017, §4).)

In general, finding the function in a convex set of probability functions which has maximum entropy is a computationally hard optimisation problem (Paris, 1994, Chapter 10). Indeed, this has been viewed as a criticism of the maximum entropy approach (Pearl, 1988, p. 463). In this paper, we employ Bayesian networks to reduce the dimension of the problem in typical cases, and thereby reduce its complexity. A Bayesian net representation of the probability function $P^\dagger$ which is motivated by OBE is called an *Objective Bayesian Net* or OBN (Williamson, 2005b). In this paper we develop a general algorithm, OBN-cDS, which generates an OBN and does so efficiently in typical cases, and we show that this algorithm is preferable to a brute-force approach to entropy maximisation. Furthermore, we explore particular situations in which one can generate an OBN faster than is possible even by OBN-cDS: this leads to algorithms OBN-2cDS and OBN-ccDS that are tailored to these situations.

In the next section, we briefly review related work. In Section 3, we present the general algorithm for constructing an OBN, OBN-cDS, and we discuss its complexity in Section 4. We then present algorithms which run faster but are only applicable in particular situations. In Section 5 and Section 6, we consider cases in which we can find an OBN without solving any optimisation problem at all, by determining the OBN instead directly from the datasets: in Section 5 we consider the two dataset case, which we generalise to *centred* datasets in Section 6. Section 7 presents cases in which the OBN cannot be inferred directly from the data but can be constructed by solving an algebraic problem—again, without the need for numerical optimisation. In Section 8, we delineate a class of scenarios in which one can construct an OBN by solving relatively few independent optimisation problems and we also give a graph-theoretical characterisation of this class.

This contribution adds to the state-of-the-art in the following ways: (a) it develops the OBE approach to data integration, (b) it shows how Bayesian net algorithms can be used to determine a maximum entropy probability function more efficiently, (c) it explores algebraic means to solve

1. There is a sense in which a collection of datasets can never be inconsistent—after all, they are just observations. However, inconsistencies can indeed arise between their measured frequency distributions, because these are assumed to be marginal distributions of some joint data-generating distribution, defined over the domain V as a whole. Thus, the datasets are inconsistent when they impose mutually contradictory constraints on this distribution, i.e., when no such joint distribution exists (cf. Remark 28).

   Note that consistency is a stronger condition than the requirement that the measured marginal distributions agree on variables in common. Due to different sample sizes, sampling bias and stochastic noise, real world datasets are rarely consistent, even in this weaker sense. In real world applications, agreement on joint domains may be achieved by taking weighted means of measured marginals with weights equal to the proportions of observations in the different datasets, or via more sophisticated meta-analyses.

the maximum entropy optimisation problem and (d) it provides philosophical underpinnings for a particular kind of 'statistical matching' technique (see below).

2. Related Work

This paper rests on three main strands of previous research.

First, we build on previous research which demonstrates the importance of maximising entropy for determining a rational belief function, i.e., for OBE. This connection was originally made by Jaynes (1957) and was defended by Tribus (1969); Rosenkrantz (1977); Paris and Vencovská (1997, 1990); Williamson (2010); Grove et al. (1994) and Landes and Williamson (2013, 2015) among others. This first strand of previous research establishes the need for computing the maximum entropy function. The importance of the maximum entropy approach to AI has been explored by, among others, Adamčík (2016); Paris (2014); Caticha (2014, 2013); Thimm et al. (2010); Balasubramanian (2005); Halpern and Koller (2004); Kern-Isberner (1998); Kern-Isberner and Rödder (2004); Kern-Isberner and Lukasiewicz (2004); Berger et al. (1996).

Second, while several steps have been taken to render determining the maximum entropy function more efficient (e.g., Cheeseman, 1983; Goldman, 1987; Goldman and Rivest, 1988; Ormoneit and White, 1999; Balestrino et al., 2006; Abramov, 2010; Chen et al., 2010; Landes and Williamson, 2016), this is a computationally hard problem, as demonstrated in the context of OBE by Maung and Paris (1990). For example, if P$\neq$NP then it is not possible to approximate the maximum entropy function in polynomial time on a deterministic Turing machine (Paris, 1994, Theorem 10.6). This casts doubt on whether one can hope to approximate the maximum entropy function in problems with large numbers of variables—these problems occur regularly in systems medicine, for instance.

Third, the extensive literature on Bayesian networks demonstrates that, although probabilistic inference is NP-complete in the worst case, one can employ Bayesian nets to reason much more efficiently in *typical* cases (see, e.g., Pearl, 1988; Neapolitan, 1990). Thus, Williamson (2005a,b) developed the OBN approach of using Bayesian nets to represent maximum entropy probability functions, in order to render inference more tractable in typical cases, if not in the worst case. This was applied to systems medicine by Nagl et al. (2008). This paper further develops the OBN approach: it uses Bayesian nets to render maximum entropy methods more efficient in typical cases.

Some other strands of work, which might be thought to be closely related, actually tackle rather different problems. For example, Paris (2005) considers the use of maximum entropy to determine conditional probabilities that are missing from a causal Bayesian network. In contrast, here we use non-causal Bayesian networks to represent maximum entropy distributions. Lukasiewicz (2000) applies the principle of maximum entropy to select a unique joint probability distribution from the set of all joint probability distributions specified by a credal network. (A credal network can be thought of as a generalisation of a Bayesian network with intervals of probabilities rather than precise probabilities.) In contrast, we seek a Bayesian network that represents such a joint distribution. Meta-analysis is a statistical method which seeks to combine findings from different studies (Borenstein et al., 2009). However, whereas meta-analysis typically seeks the overall effect of a treatment, i.e., to assess a single causal relationship, here the task is to determine and represent a joint probability distribution. Another strand of research is concerned with inferring causal relationships, rather than rational degrees of belief, from multiple datasets (Danks, 2002; Danks et al., 2008; Triantafillou et al., 2010; Tsamardinos et al., 2012; Tillman and Spirtes, 2011; Tillman and

Eberhardt, 2014; Triantafillou and Tsamardinos, 2015; Bareinboim and Pearl, 2016; Mayo-Wilson, 2019). Discovery of causal relationships is a very different problem to the one we tackle here.

Statistical methods for handling missing data are more closely related to the present work: if one concatenates several datasets that measure overlapping sets of variables, then one is left with a single dataset with lots of missing values. Little and Rubin (2014, p. 19-20) identify four types of methods for handling missing data. First, one might discard those lines of the concatenated dataset which contain missing values. This would be inadequate in our context, where all the lines will contain missing values. The second method involves the use of weighting procedures; these apply to values that are missing in survey responses, which is not the case here, since we presume that all attempted measurements were successful. Third, imputation-based procedures attempt to fill in missing data by direct extrapolation from the values that are actually present. The fourth method is to build a model and sample from that model to fill in missing data. In our context, these latter two methods could play the following role: first one can apply such a method to fill in the missing data; then one can apply standard methods for learning a Bayesian network from complete data to represent a joint distribution on the domain as a whole. However, adopting this sort of two-step procedure is undesirable because the first step—filling in the missing data—is unreliable when a large proportion of values are missing, as is the case in our context. Our approach avoids this error-prone step: in this paper we learn a model of the joint distribution directly from the data that is present.

The present work can also be viewed as an instance of *statistical matching*: the problem of integrating datasets which measure different sets of variables and which sample different individuals.[2] D'Orazio et al. (2006) distinguish two approaches to statistical matching. The 'micro' approach aims to produce a single dataset covering the domain as a whole from all the individual datasets that measure subsets of variables, by synthetically producing extra data. This is the missing-data approach outlined above. On the other hand, the 'macro' approach to statistical matching uses the given datasets to produce a joint distribution over the domain as a whole. The approach of this paper is a 'macro' approach: here the joint distribution is the maximum entropy distribution. This is an approach not considered by D'Orazio et al. (2006) or Vantaggi (2008), who focus on the special case of two datasets. Endres and Augustin (2016) develop an approach to statistical matching that builds a joint distribution by assuming various conditional probabilistic independencies.

However, in applications with many datasets it can be hard to justify conditional probabilistic independence assumptions between sets of variables measured by different datasets. The maximal entropy approach avoids this problem because it does not make any such independence assumptions. On the other hand, the use of the maximum entropy function does guarantee that certain probabilistic independencies obtain and the OBN exploits this feature to reduce the dimension of the entropy maximisation problem. The maximum entropy approach can thus be thought of as providing a principled way of avoiding ad hoc independence assumptions.

3. The General Algorithm OBN-cDS

In this section, we present a general algorithm for finding an OBN, given evidence that takes the form of consistent datasets $DS_1, DS_2, \ldots, DS_h$. The algorithm is called OBN-cDS for its ability to construct an OBN from consistent datasets.

2. See Nielsen (2016); Schleicher et al. (2020) for reviews of the applicability of different matching approaches to concrete problems.

3.1 Notation and Motivation

Notation is aligned with Williamson (2005a).

Definition 1 (Variables and Assignments)**.** Variables are denoted by (sometimes subscripted) upper case letters A, B, C, the set of all variables by $V \neq \emptyset$. V has size $n \geq 2$ and variables have arbitrary finite arity greater or equal than two. Let $\#V$ denote the number of states of V. In case all variables are binary, we have $2^n = \#V$. We denote by V_i the set of variables measured by dataset DS_i.

An assignment of values to a set of variables $U \subseteq V$ is written as $u@U$ and the value of variable $A \in U$ under assignment u is denoted by a^u. We also write b^u for as set of variables $B \subseteq U$, meaning that all variables in B are assigned a value according to u. Sometimes we need to modify an assignment v. We denote by sv for $s@S$ the assignment obtained from v by assigning all variables in S their value according to s. For example for $s@S$ and $t@S$ it holds that $st = s$.

Definition 2 (Probabilities and Entropy)**.** A probability function P maps each assignment $v@V$ to $[0, 1]$ such that $\sum_{v@V} P(v) = 1$ and for all $U \subset V$ all $u@U$ and all $A \in V \setminus U$ it holds that $\sum_{v@\{A\}} P(ua^v) = P(u) \in [0, 1]$. The entropy $H(P)$ of a probability function P is defined as $H(P) := -\sum_{v@V} P(v) \log P(v)$, where $0 \cdot \log 0 = 0$. The base of the logarithm is strictly greater than 1 but otherwise arbitrary; for concreteness we take the logarithm to be the natural logarithm in this paper.

Motivation & Example. Consider datasets DS_1, DS_2, DS_3 which measure sets of binary variables $\{A_0, A_1\}$, $\{A_0, A_2\}$ and $\{A_1, A_2\}$. From the datasets we obtain measured frequency distributions P_1^*, P_2^*, P_3^*, which measure respectively $\{A_0, A_1\}$, $\{A_0, A_2\}$ and $\{A_1, A_2\}$. The consistency condition (cf. Footnote 1 and Remark 28) ensures that P_1^* and P_2^* agree on $\{A_0\}$, P_1^* and P_3^* agree on $\{A_1\}$ and P_2^* and P_3^* agree on $\{A_2\}$. Note that none of the P_i^* is defined on the set of all variables $V = \{A_1, A_2, A_3\}$ and hence the problem arises as to how to determine beliefs on the entire domain of interest (we return to the example of Table 1 and Example 31).

In general, there are uncountably many probability functions defined on V which agree with all P_i^* simultaneously. This paper is concerned with efficiently computing the unique probability function $P^\dagger$, from all the probability functions defined on V that agree with each P_i^*, that has maximum entropy.[3] The full formal problem to determine an OBN is stated in Section 3.3.5.

The Optimisation Problem. To find the probability function $P^\dagger$ defined on the entire domain of interest that has maximal entropy and agrees with all measured marginal probability distributions one needs to find the solution to the following optimisation problem:

$$\begin{aligned}
\text{maximise: } & H(P) = -\sum_{v@\{V\}} P(v) \log(P(v)) \\
\text{subject to: } & P(v) \geq 0 \text{ for all } v@\{V\} \\
& \sum_{v@\{V\}} P(v) = 1 \\
& P(s) = P_i^*(s) \text{ for all } 1 \leq i \leq h \text{ and all } s@\{V_i\} \ .
\end{aligned}$$

The number of unknowns in the optimisation problem, i.e., the number of probabilities $P(v)$, grows exponentially in the number of measured variables. If all variables are binary, then the number of

3. Uniqueness follows from the entropy function being strictly concave and the feasible region being convex.

Table 1: Example of three consistent datasets where every variable is measured by two datasets. The black cells indicate variables not measured by the corresponding dataset.

		Variables measured in Dataset 1	Variables measured in Dataset 2	
	Observation	Sex	Treatment Outcome	Age
Observations of Dataset 1	Patient 1	Male	survived	
	Patient 2	Female	died	
	Patient 3	Female	survived	
Observations of Dataset 2	Patient 4		survived	$60+$ years
	Patient 5		survived	$60+$ years
	Patient 6		died	$40-59$ years
Observations of Dataset 3	Patient 7	Female		$60+$ years
	Patient 8	Male		$60+$ years
	Patient 9	Female		$40-59$ years
		Variable measured in Dataset 3		Variable measured in Dataset 3

unknowns is equal to $2^n - 1$. Solving this optimisation problem by 'brute force'—i.e., without efforts to reduce its complexity—is simply not feasible for large n. Indeed, no method will be feasible for all problems where n is large: if RP$\neq$NP then there is no random Turing machine that can probably approximate the correct solution in polynomial time—see, e.g., Paris (1994, Theorem 10.7). Hence, there is a need for an approach to compute $P^\dagger$ that does not work by brute force and that is computationally tractable in many natural cases, if not the worst case.

3.2 Overview of OBN-cDS

The algorithm that we present relies on the ability to learn Markov network structures that represent the conditional independencies satisfied by the marginal probability distributions measured in the datasets.

Markov Network Structure Learning. There are numerous algorithms for learning Markov and Bayesian networks that approximate the measured probabilities of a single dataset, P_i^*. See Neapolitan (2003) for general background. Our general algorithm presupposes the ability to learn, from a given dataset, the structure of a Markov network representation of the dataset distribution. Such a structure consists in an undirected graph on the set V_i of vertices that represents conditional probabilistic independencies satisfied by P_i^* by means of graph separation: if sets X and Y of variables are separated by a set Z of variables in the graph (i.e., every pathway from a variable in X to a variable in Y proceeds via a variable in Z) then P_i^* renders X and Y to be probabilistically independent, conditional on Z. Bromberg et al. (2009), for example, provide an algorithm for efficient Markov net structure discovery.

It is also possible to first learn a Bayesian net structure and then convert that structure to a Markov net structure. A Bayesian net consists in a directed acyclic graph (DAG) on the set V_i of vertices (the 'structure' of the net) together with the probability distribution of each variable conditional on its parents in the DAG (the 'parameters' of the net). These are related by an assumption known as the Markov Condition, which says that for all $A \in V_i$, A is probabilistically independent

of its non-descendants in the DAG conditional on its parents. Tsamardinos et al. (2006) provide an algorithm for learning a Bayesian net structure, for example. This structure can be converted to a Markov net structure by 'marrying' any unmarried parents in the DAG (i.e., adding an undirected edge between those two variables) and dropping the orientations of the remaining edges.

Pseudo Code of OBN-cDS. We now show how to turn these DAG structures into an OBN. Steps 1–2 construct an undirected graph representing the conditional independence structure of the maximum entropy function, $P^\dagger$. Steps 3–5 then translate this into a Bayesian network. The correctness of the algorithm is demonstrated below.

Input: consistent datasets $DS_1, \ldots, DS_h$.
1) For all i learn a Markov network structure $\mathcal{G}_i$ from DS_i representing independences of P_i^*.
2) Set overarching undirected graph $\mathcal{G}$ as the union of the $\mathcal{G}_i$.
3) Compute a minimal triangulation $\mathcal{G}^T$ of $\mathcal{G}$.
4) Orientate $\mathcal{G}^T$ to give DAG $\mathcal{H}$.
5) For each vertex in $\mathcal{H}$, determine its probability distribution conditional on its parents:
   a) For all vertices for which there exists a dataset which measures this vertex and all its parents determine conditional probabilities as described in Section 3.3.3.
   b) For all other vertices determine conditional probabilities by solving the optimisation problem specified below.
Output: Objective Bayesian Net with DAG $\mathcal{H}$ and conditional probabilities as determined in Step 5.

Next we will specify the algorithm in more detail and specify regularity conditions under which the algorithm is guaranteed to succeed. In Section 4 we will discuss how to further improve the efficiency of the algorithm: Step 3 is addressed in Sections 4.1 and 4.2; Step 4 is considered in Sections 4.1 and 4.4.

3.3 OBN-cDS in Detail

We now explain the general algorithm OBN-cDS in more detail.

3.3.1 DETERMINING THE STRUCTURE OF THE OBJECTIVE BAYESIAN NET

Step 1. For every dataset DS_i we learn the structure $\mathcal{G}_i$ of a Markov network which represents conditional probabilistic independencies in the distribution measured by the dataset, P_i^*.

Step 2. We form an undirected graph $\mathcal{G}$ by joining the $\mathcal{G}_i$: take the variables in $V = \bigcup_i V_i$ as vertices and connect every pair of vertices that are connected in some $\mathcal{G}_i$.

Step 3. Compute a minimal triangulation $\mathcal{G}^T$ of $\mathcal{G}$. A *triangulation* of $\mathcal{G}$ is a supergraph of $\mathcal{G}$ in which each simple cycle of length greater or equal than 4 possesses a chord. A triangulation of $\mathcal{G}$ is *minimal* if none of its proper subgraphs is a triangulation of $\mathcal{G}$. See Heggernes (2006) for more background on triangulations.

Step 4. Transform $\mathcal{G}^T$ into a DAG $\mathcal{H}$ that also represents the independence structure of $P^\dagger$. This is a standard transformation, which can be achieved as follows; see Williamson (2005a, §5.7) and Neapolitan (1990) for further discussion and details. (i) Order the vertices of $\mathcal{G}^T$ with vertex set V according to maximum cardinality search: at each step select a vertex which is adjacent to the largest number of previously numbered vertices. (ii) Let $D_1, \ldots, D_l$ be the cliques of $\mathcal{G}^T$, ordered according to the highest labelled vertex. (iii) Let $E_j := D_j \cap (\bigcup_{i=1}^{j-1} D_i)$ and $F_j := D_j \setminus E_j$. (iv)

Add an arrow from each vertex in E_j to each vertex in F_j. (v) Add further arrows to ensure there is an arrow between each pair of vertices in D_j such that the resulting directed graph $\mathcal{H}$ is acyclic. Arbitrarily break ties and arbitrarily make unconstrained choices.

We use the following procedure for inserting arrows in every connected component of $\mathcal{G}^T$ in (v). This procedure has the advantage that it is easily implementable. First, observe that every edge is in some clique. Next, note that if an orientation of the unoriented edges results in a *directed* cyclic graph, then there has to exist a clique which contains all vertices of a directed cycle. This is because all arrows between cliques originate from cliques earlier in the ordering and point to the cliques later in the ordering. So, all we need to ensure is that there is no directed cycle within any clique. For all $1 \leq i \leq l$ and all edges in clique D_i which are not orientated in (iv), these are the edges between vertices in the F_i and the edges between vertices in D_i. Now orient the edge such that it originates from the vertex which comes prior to the vertex to which it points towards in the maximum cardinality ordering. We use this way of breaking ties in our Matlab implementation, which is described in §3.4. As a consequence of this procedure, every connected component of $\mathcal{H}$ has a unique root, i.e., a variable which does not have a parent.

Before proceeding to Step 5, we provide a result which provides conditions under which Steps 1-4 are guaranteed to succeed. If the Markov network structure learning in Step 1 is successful then Steps 2-4 are guaranteed to produce a Bayesian network structure that can be used to represent the maximum entropy function $P^\dagger$:

Theorem 3. *If each dataset distribution P_i^* satisfies the conditional probabilistic independence relationships represented by $\mathcal{G}_i$, for $i = 1, \ldots, h$, then the maximum entropy function $P^\dagger$, from all those probability functions that match the P_i^*, satisfies the conditional probabilistic independence relationships represented by $\mathcal{H}$.*

Proof: The key elements of the proof are developed in Appendix A.

The main fact that underpins the algorithm is that separation in the graph $\mathcal{G}$ produced by Step 2 represents probabilistic independencies satisfied by $P^\dagger$: for arbitrary subsets X, Y, Z of variables, if Z separates X from Y in $\mathcal{G}$ then $X \perp\!\!\!\perp_{P^\dagger} Y | Z$. This is Theorem 34 of Appendix A, which appeals to results of Williamson (2002, 2005a).

Steps 3 and 4 are a fairly standard way of transforming an undirected graph which represents independencies by means of separation, into a directed acyclic graph which represents independencies by means of D-separation, as defined by Pearl (1988, §3.3.1). That the independencies represented by $\mathcal{H}$ do indeed hold for the maximum entropy function $P^\dagger$ is Theorem 35 of Appendix A. ■

Note that $P^\dagger$ may satisfy further probabilistic independence relationships, not captured by $\mathcal{H}$. Indeed, it may be that there is no directed acyclic graph that captures all the independencies satisfied by $P^\dagger$. This does not matter for our purpose here, which is to exploit those independencies that are represented by $\mathcal{H}$ to reduce the dimension of the entropy maximisation problem.

We next turn to Step 5: determining the conditional probabilities that are the parameters of the objective Bayesian net.

3.3.2 Determining the Parameters of the Objective Bayesian Net

The conditional probabilities in the OBN can be obtained by solving an optimisation problem. These can be found by computing the probability function, from all those that agree with the dataset distributions P_i^*, that has maximum entropy. The objective function to be maximised is the entropy

of a probability function, $H(P)$. Probability functions satisfying the probabilistic independences specified by the structure (the DAG) $\mathcal{H}$ of our Bayesian network can be conveniently represented as follows:

Definition 4 (Objective Function and y-parameters, Williamson (2005a) (p. 93))**.** Let Anc_i be the ancestors of A_i in $\mathcal{H}$; $Anc_i' := \{A_i\} \cup Anc_i$; and Par_i be the set of parents of A_i in $\mathcal{H}$. The entropy of a probability function P that satisfies the probabilistic independencies represented by $\mathcal{H}$ is:

$$\begin{aligned} H(P) &:= -\sum_{v@V} P(v)\log(P(v)) \\ &= -\sum_{i=1}^{n}\sum_{v@Anc_i'}\Big(\prod_{A_j\in Anc_i'} P(a_j^v|Par_j^v)\Big)\log P(a_i^v|Par_i^v) \\ &= -\sum_{i=1}^{n}\sum_{v@Anc_i'}\Big(\prod_{A_j\in Anc_i'} y_j^v\Big)\log y_i^v. \end{aligned} \tag{1}$$

Here each y_j^v is a parameter which denotes $P(a_j^v|Par_j^v)$. These y-parameters are the unknowns to be determined by the solution of the optimisation problem. Note that y_j^v only depends on values assigned to A_j and all its parents.

3.3.3 VARIABLES OF THE OPTIMISATION PROBLEM

If a variable A_i and all its parents are measured in the same dataset, DS_k say, then for all $v@Par_i \cup \{A_i\}$, $P_k^*(v)$ can be obtained from DS_k. Since $P^\dagger$ and P_k^* have to agree, such parameters can typically be computed *without* solving an optimisation problem.[4] The conditional probability $y_j^v = P(a_j^v|Par_j^v)$ can be determined from the dataset DS_k itself, by dividing the number of observations in DS_k in which $a_i^v Par_i^v$ holds by the number of observations in which Par_i^v holds.[5]

If Par_i^v has no observed instances in any dataset which measures A_i and all its parents, then the parameter $y_i^v = P(a_i^v|Par_i^v)$ cannot be determined from the dataset, because the measured frequency is $0/0$. In such a case, this parameter will not contribute to any probabilities calculated from the OBN, since all of its occurrences in (1) are in terms which are multiplied by 0. Nevertheless, it needs to be given a value in order to fully specify the net. In such a situation, we set $y_i^v = 1/k_i$, where k_i is the number of possible values of variable A_i, because these are the maximum entropy values in the absence of any relevant evidence, and thus the natural objective Bayesian solution.

Next we turn to the situation in which there is no dataset that measures a variable and all its parents.

Definition 5 (Under-determined Variable)**.** We call a variable $A \in V$ *under-determined*, if and only if there is no dataset jointly measuring A and all its parents. A variable $A \in V$ is either under-

4. If there exists a further dataset DS_l which measures A_i and all its parents, then $P_k^*(v) = P_l^*(v)$ because datasets are assumed to be consistent. Hence, $P^\dagger(v)$ can be uniquely determined as long as there exists at least one dataset which measures A_i and all its parents, modulo the qualification below.
5. An alternative objective Bayesian approach to determining these conditional probabilities involves adopting an objective prior distribution over these probability parameters and then updating this prior in the light of the data. This alternative strategy introduces further complexities but is a natural approach to pursue when the datasets are too small to provide acceptable estimates of the underlying frequency distribution. In this paper, it is assumed that the datasets are large enough for their distributions to be used as frequency estimates.

determined or we can determine the values of the y_i^v as described above, in which case the variable is said to be *fully determined*.

In particular, every under-determined variable has at least one parent. So, the y-parameters corresponding to variables that are roots of the OBN structure $\mathcal{H}$ can be determined directly.

If A has two or more parents, arrows from those parents to A are said to *collide* at A, and A is called a *collider* variable. In the simplest such case A has precisely two parents—see Figure 1. The variable A is under-determined, if and only if there is no dataset which jointly measures A and both its parents, A_1 and A_2.

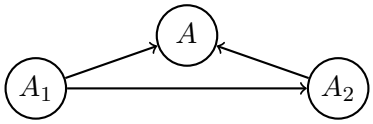


Figure 1: Collider variable A with parents A_1, A_2.

If it were the case that every arrow in $\mathcal{H}$ connects two variables which are jointly measured in some dataset, then no variable with only one parent would be under-determined. As we shall now see, this is not always the case.

Example 6. Consider a collection of four datasets where DS_1 measures A_1, A_2, DS_2 measures A_2, A_3, DS_3 measures A_3, A_4 and DS_4 measures A_4, A_1. All variables are distinct. If the over-arching undirected graph $\mathcal{G}$ is the cycle $A_1 - A_2 - A_3 - A_4 - A_1$, then every triangulation of $\mathcal{G}$ will either contain an edge between A_1 and A_3, an edge between A_2 and A_4 or both these edges. In all these cases, there exists an edge between two variables which are not jointly measured in any dataset, see Figure 2.

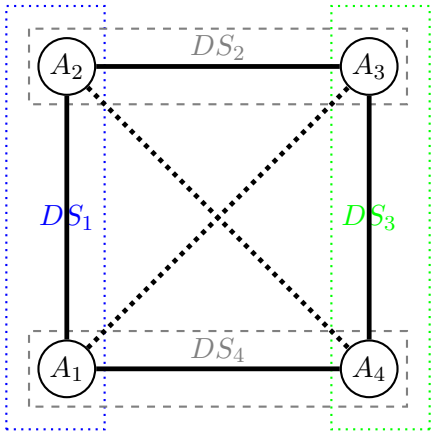


Figure 2: Every triangulation will either introduce an edge between A_1 and A_3 or an edge between A_2 and A_4 or both edges (dotted). No two variables connected by a dotted edge have been jointly measured in a single dataset.

Step 3 of OBN-cDS (triangulation) is the only step at which edges are added and where there is some free choice as to how to add these edges. Where possible, the creation of edges should be

avoided, in order to minimise the number of under-determined variables. However, as we saw in the example depicted in Figure 2 and Figure 3 this is not always possible. Where the creation of edges cannot be avoided, our approach does, in general, require the solution of an optimisation problem to compute the parameters of an OBN.

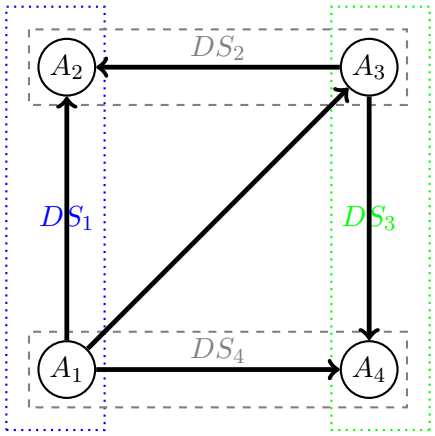


Figure 3: This orientation of the graph depicted in Figure 2 has the under-determined variable A_3 which only has the single parent A_1. Hence, variables with a single parent (i.e., variables which are not colliders) may be under-determined, too.

In Section 7, we delineate cases in which one cannot apply Step 5a to find all (conditional) probabilities, but in which it is nevertheless possible to find an OBN *without* solving an optimisation problem.

3.3.4 THE CONSTRAINTS FOR UNDER-DETERMINED VARIABLES

We now give pseudo code for determining the constraints relating to under-determined variables. Explanations and examples will follow.

Input: The DAG of OBN and the allocation of variables to the datasets measuring them.
1) Determine the subset of variables $\mathcal{U} \subset V$ such that $A_i \in \mathcal{U}$ if and only if there exists no dataset jointly measuring A_i and all its parents.
2) For all variables $A_i \in \mathcal{U}$ and all assignments $s@Par_i$ to the parents of A_i we have the following constraints: $1 = \sum_{r@\{A_i\}} P(A_i^r|Par_i^s)$ and $0 \leq P(A_i^r|Par_i^s) \leq 1$.
3) For all variables $A_i \in \mathcal{U}$, all datasets DS_k measuring A_i, all subsets $Par_{i:k}$ of parents of A_i measured by DS_k, all assignments $s@Par_{i:k}$ and all assignments $v@\{A_i\}$ we have the constraints: $P^\dagger(A_i^v|Par_{i:k}^s) = P_k^*(A_i^v|Par_{i:k}^s)$, if $P_k^*(Par_{i:k}^s) > 0$.
Output: Constraints.

Step 1 computes the set of under-determined variables. Step 2 ensures that a probability function is well defined. Step 3 ensures that $P^\dagger$ agrees with all measured marginal probability distributions P_k^*.

Suppose that dataset DS_k measures the under-determined variable A_r and a possibly empty subset of parents $Par_{r:k}$ of A_r and no other dataset measures a strict superset of $\{A_r\} \cup Par_{r:k}$.[6] Recall that all datasets are assumed to be consistent, so no conflict arises where two datasets measure the same variables. For all $v@\{A_r\} \cup Par_{r:k}$ the following constraint has to hold:

$$P^\dagger(v) = P^\dagger(a_r^v Par_{r:k}^v) = P_k^*(a_r^v Par_{r:k}^v) = P_k^*(v) \ .$$

Note that all constraints which arise from the requirement that $P^\dagger$ ought to agree with measured probabilities are of this form.

We shall now rewrite this constraint as a constraint on the y-parameters. The right hand side can be determined from DS_k as described in Section 3.3.3. Then for all $v@\{A_r\} \cup Par_{r:k}$, assuming that $P(Par_{r:k}^v) > 0$ and hence the conditional probabilities are well-defined, it must hold that:

$$\begin{aligned} P^\dagger(a_r^v Par_{r:k}^v) &= \sum_{s@Anc_r \setminus Par_{r:k}} P^\dagger(a_r^v Par_{r:k}^v s) \\ &= \sum_{s@Anc_r \setminus Par_{r:k}} P^\dagger(a_r^v | Par_r^{sv}) \prod_{A_i \in Anc_r} P^\dagger(a_i^{sv} | Par_i^{sv}) \\ &= \sum_{s@Anc_r \setminus Par_{r:k}} \prod_{A_i \in Anc_r'} P^\dagger(a_i^{sv} | Par_i^{sv}) \\ &= \sum_{s@Anc_r \setminus Par_{r:k}} \prod_{A_i \in Anc_r'} y_i^{sv} \ . \end{aligned} \tag{2}$$

We also need to ensure that we restrict the feasible region to *probability* functions. Hence, the constraint that no probability is less than zero and greater than one does, of course, also apply. Furthermore, it must also be the case that for all variables and all assignments v of their parents that conditional probabilities add up to 1:

$$\sum_{g@A_r} P^\dagger(a_r^g | Par_r^v) = \sum_{g@A_r} y_r^{gv} = 1 \ .$$

There are no further constraints to satisfy. All constraints at which we arrive in this fashion are multi-linear in the y-parameters.

Of course, not all y-parameters are unknown optimisation parameters in our approach. The value of some subset of the y-parameters may be established by Step 5a, as explained above and as illustrated by the following example.

Example 7. Let us consider the DAG in Figure 3 with binary variables and determine constraints for A_4. For the assignment $v@\{A_3, A_4\}$ making A_3 and A_4 true, we have the following constraint

$$\begin{aligned} P^*(a_3 a_4) =& P^\dagger(a_3 a_4) \\ =& P^\dagger(a_1 a_3 a_4) + P^\dagger(\overline{a}_1 a_3 a_4) \\ =& y_4^{a_4 a_1 a_3} P^\dagger(a_3 | a_1) P^\dagger(a_1) + y_4^{a_4 \overline{a}_1 a_3} P^\dagger(a_3 | \overline{a}_1) P^\dagger(\overline{a}_1) \\ =& y_4^{a_4 a_1 a_3} y_3^{a_3 a_1} y_1^{a_1} + y_4^{a_4 \overline{a}_1 a_3} y_3^{a_3 \overline{a}_1} y_1^{\overline{a}_1} \end{aligned}$$

6. Figure 3 is an example in which no dataset measures an under-determined variable and at least one of its parents. In such a case, constraints arise from datasets which measure this under-determined variable.

$$=y_4^{a_4a_1a_3}y_3^{a_3a_1}P^*(a_1)+y_4^{a_4\overline{a}_1a_3}y_3^{a_3\overline{a}_1}(1-P^*(a_1))\ .$$

For the assignment $v'@\{A_3,A_4\}$ making A_3 true and A_4 false, we have the following constraint

$$\begin{aligned}P^*(a_3\overline{a}_4)=&P^\dagger(a_3\overline{a}_4)\\=&P^\dagger(a_1a_3\overline{a}_4)+P^\dagger(\overline{a}_1a_3\overline{a}_4)\\=&y_4^{\overline{a}_4a_1a_3}P^\dagger(a_3|a_1)P^\dagger(a_1)+y_4^{\overline{a}_4\overline{a}_1a_3}P^\dagger(a_3|\overline{a}_1)P^\dagger(\overline{a}_1)\\=&y_4^{\overline{a}_4a_1a_3}y_3^{a_3a_1}y_1^{a_1}+y_4^{a_4\overline{a}_1a_3}y_3^{a_3\overline{a}_1}y_1^{\overline{a}_1}\\=&(1-y_4^{a_4a_1a_3})y_3^{a_3a_1}P^*(a_1)+(1-y_4^{a_4\overline{a}_1a_3})y_3^{a_3\overline{a}_1}(1-P^*(a_1))\ .\end{aligned}$$

For the assignment $v''@\{A_3,A_4\}$ making A_3 false and A_4 true, we have the following constraint

$$\begin{aligned}P^*(\overline{a}_3a_4)=&P^\dagger(\overline{a}_3a_4)\\=&P^\dagger(a_1\overline{a}_3a_4)+P^\dagger(\overline{a}_1\overline{a}_3a_4)\\=&y_4^{a_4a_1\overline{a}_3}P^\dagger(\overline{a}_3|a_1)P^\dagger(a_1)+y_4^{a_4\overline{a}_1\overline{a}_3}P^\dagger(\overline{a}_3|\overline{a}_1)P^\dagger(\overline{a}_1)\\=&y_4^{a_4a_1\overline{a}_3}y_3^{\overline{a}_3a_1}y_1^{a_1}+y_4^{a_4\overline{a}_1\overline{a}_3}y_3^{\overline{a}_3\overline{a}_1}y_1^{\overline{a}_1}\\=&y_4^{a_4a_1\overline{a}_3}(1-y_3^{a_3a_1})P^*(a_1)+y_4^{a_4\overline{a}_1\overline{a}_3}(1-y_3^{a_3\overline{a}_1})(1-P^*(a_1))\ .\end{aligned}$$

For the assignment $v'''@\{A_3,A_4\}$ making A_3 and A_4 false, we have the following constraint

$$\begin{aligned}P^*(\overline{a}_3\overline{a}_4)=&P^\dagger(\overline{a}_3\overline{a}_4)\\=&P^\dagger(a_1\overline{a}_3\overline{a}_4)+P^\dagger(\overline{a}_1\overline{a}_3\overline{a}_4)\\=&y_4^{\overline{a}_4a_1\overline{a}_3}P^\dagger(\overline{a}_3|a_1)P^\dagger(a_1)+y_4^{\overline{a}_4\overline{a}_1\overline{a}_3}P^\dagger(\overline{a}_3|\overline{a}_1)P^\dagger(\overline{a}_1)\\=&y_4^{\overline{a}_4a_1\overline{a}_3}y_3^{\overline{a}_3a_1}y_1^{a_1}+y_4^{\overline{a}_4\overline{a}_1\overline{a}_3}y_3^{\overline{a}_3\overline{a}_1}y_1^{\overline{a}_1}\\=&(1-y_4^{a_4a_1\overline{a}_3})(1-y_3^{a_3a_1})P^*(a_1)+(1-y_4^{a_4\overline{a}_1\overline{a}_3})(1-y_3^{a_3\overline{a}_1})(1-P^*(a_1))\ .\end{aligned}$$

Similarly, four constraints arise for the assignments $s@\{A_1,A_4\}$.

3.3.5 THE OPTIMISATION PROBLEM

After determining the y-parameters of fully determined variables via Step 5a and collecting the set of under-determined variables by letting

$$\mathcal{U}:=\{1\le i\le n\ :\ A_i \text{ is under-determined}\},\tag{3}$$

we can state the optimisation problem as follows:

$$\text{maximise: } -\sum_{i=1}^{n}\sum_{v@Anc_i'}\Big(\prod_{A_j\in Anc_i'}y_j^v\Big)\log y_i^v \tag{4}$$

$$\text{subject to: } y_i^v\ge 0 \text{ for all } i\in\mathcal{U} \tag{5}$$

$$\sum_{g@\{A_i\}}y_i^{gv}=1 \text{ for all } i\in\mathcal{U} \tag{6}$$

$$\sum_{s@Anc_i \setminus Par_{i:k}} \prod_{A_j \in Anc'_i} y_j^{sv} = P_k^*(a_i^v Par_{i:k}^v) \text{ for all } i \in \mathcal{U} \text{ and}$$

all maximal jointly measured sets of the form $\{A_i\} \cup Par_{i:k}$. (7)

Note that the first two constraints jointly entail that all y-parameters are less or equal than 1.

Proposition 8 (Agreement with Measured Marginals)**.** *All conditional probabilities of the OBN agree with all measured marginal probability distributions P_k^* defined on V_k.*

Proof: The (conditional) probabilities according to $P^\dagger$ of each fully determined variable in the DAG $\mathcal{H}$ are equal to the measured marginals by construction (Step 5a).

Next consider the conditional probabilities according to $P^\dagger$ of an under-determined variable A_i in the DAG $\mathcal{H}$. For all valuations v the conditional probability of $P^\dagger(a_i^v|par_{i:k}^v)$, where A_i is measured by DS_k, is equal to the measured marginals by construction, $P_k^*(a_i^v|par_{i:k}^v)$, as explained in Section 3.3.4. ■

Note that this does not mean that the distribution determined by the OBN is guaranteed to agree with the measured marginal probability distributions, since the structure learning algorithm employed in Step 1 of the algorithm only approximates the measured marginals. If the structure learning approximation finds too few edges, then the OBN distribution is likely to have erroneous independencies and thus likely to fail to agree with the dataset distributions. Nevertheless we do have the following guarantee.

Proposition 9 (Correctness of the Optimisation)**.** *The OBN represents the probability function with maximal entropy given the evidential constraints and the independence structure determined by the DAG $\mathcal{H}$.*

Proof: Optimisation is carried out with respect to the evidential constraints (Section 3.3.4) ensuring that all probability functions in the feasible region agree with all evidential constraints. Since optimisation only provides values for under-determined variables and we impose no further constraints, the independence structure of the DAG $\mathcal{H}$ is preserved. ■

This leads to a guarantee of correctness of the algorithm as a whole, conditional on Step 1 being successful.

Theorem 10 (Correctness of OBN-cDS)**.** *If each dataset distribution P_i^* satisfies the conditional probabilistic independence relationships represented by $\mathcal{G}_i$, for $i = 1, \ldots, h$, then OBN-cDS outputs an OBN that represents the probability function $P^\dagger$, from all those that agree with all measured marginal probability distributions P_k^*, that has maximal entropy.*

Proof: $P^\dagger$ satisfies the conditional independence relationships represented by the graphical structure $\mathcal{H}$ of the OBN output by OBN-cDS (Theorem 3). The parameters of the OBN ensure that the evidential constraints are satisfied and that entropy is maximised (Proposition 9). Thus, the OBN represents $P^\dagger$, as required. ■

In concrete applications, the condition of Theorem 10 may not hold: the learned structures $\mathcal{G}_i$ will often only approximate the independence structures of the dataset distributions. In which case, the Bayesian network output by OBN-cDS will provide an approximation of the maximal entropy function $P^\dagger$.

Example 11. Let us again consider the situation in Figure 3 and let $v@\{A_1, A_2, A_3, A_4\}$. Then the conditional probability $P^\dagger(a_3^v|a_1^v)$ is determined by the solution of the optimisation problem as is the conditional probability $P^\dagger(a_4^v|a_1^v a_3^v)$ under the constraint that $P^\dagger(a_4^v|a_3^v) = P_3^*(a_4^v|a_3^v)$.

If the situation is instead as in Figure 4, we can either either use a minimal triangulation (Step 3) connecting A_1 and A_3 or A_2 and A_4. In the former case, we can enumerate the variables in the standard way (A_1, A_2, A_3, A_4); arrows then originate from variables earlier in the enumeration and point towards variables appearing later in the enumeration. All variables are fully determined. The (conditional) dependencies according to $P^\dagger$ among the A_1, A_2, A_3 and among the A_1, A_3, A_4 are hence exactly as in the measured marginal frequencies. Assuming that the (conditional) independence A_1 of A_3 given A_2 in $\mathcal{G}_1$ was learned correctly, determining the conditional probability of A_3 given A_1 and A_2 from P_1^* will give conditional probabilities for $P^\dagger$ of A_3 such that A_3 is independent of A_1 given A_2. This means that the edge connecting A_1 and A_3 could be dropped from our OBN even though our general algorithm for creating the DAG $\mathcal{H}$ contains an arrow originating from A_1 and pointing to A_3.

In the latter case either A_2 or A_4 is under-determined, since A_2 and A_4 are not jointly measured. Beginning our enumeration with A_1, A_2 we are then forced to complete the enumeration with A_4, A_3 (Step 4), see the right-hand side of Figure 4. According to the learned structure $\mathcal{G}_1$, A_1 and A_3 are independent given A_2. According to the learned structure $\mathcal{G}_2$, A_1 and A_3 are independent given A_4. In general, neither are A_1 and A_3 independent given A_2 nor are they independent given A_4 according to our OBN; they are only independent according to our OBN given A_2, A_4.

We come back to the two dataset case in Section 5.

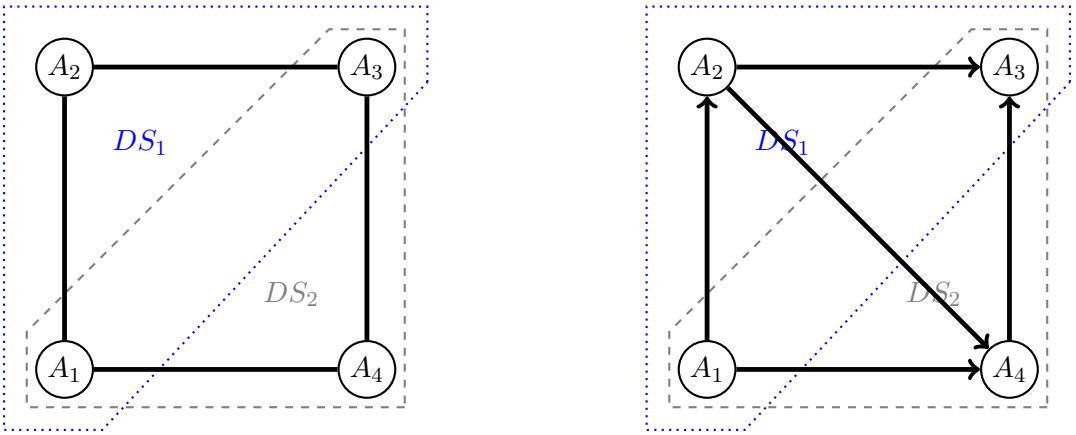


Figure 4: Left: a minimal triangulation this structure either connects A_1 to A_3 or A_2 to A_4. Right: this triangulation yields the under-determined variable A_4.

3.4 Proof of Concept

In order to compare OBN-cDS with the brute-force approach, we implemented both approaches in Matlab. In this section we describe this proof of concept implementation. The primary aim of this implementation was to test the extent to which OBN-cDS reduces the size of the optimisation problem, in comparison to the brute-force approach. Results confirm that OBN-cDS does indeed enable OBN construction in situations in which the brute-force method is not feasible due to its

computational complexity. While efficiency of implementation was not a primary goal, some results relating to run times are also reported below. Further potential efficiency improvements will be discussed in subsequent sections.

Set up. In order to test our hypothesis we require consistent datasets. We thus created 3 datasets as follows. We first specified a Bayesian network on a set of binary variables $A_1, A_2, \ldots, A_n$. We used a density parameter d^* representing the number of arrows in the DAG. For fixed d^*, we inserted arrows uniformly at random between variables. The orientation was fixed by directing each arrow from the variable enumerated first to that with the greater index. This ensured that the directed graph is acyclic. Unconditional and conditional probabilities of all variables were assigned uniformly at random in the interval $[0, 1]$. A single dataset was created by sampling from this Bayesian network. This Bayesian network was hidden for the remainder of computations.

Next, we assigned every variable in V to a non-empty set of datasets in which it was measured. To obtain computationally interesting problems we fixed A_1 to be measured by DS_1 and DS_2; A_2 to be measured by DS_1 and DS_3; and A_3 to be measured by DS_2 and DS_3. The other variables were assigned uniformly at random to the seven non-empty subsets of the 3 datasets. We then created three clones of the sampled dataset. In all three clones we deleted the columns corresponding to measurements of variables not measured by this dataset. In this way we arrived at our collection of three consistent datasets. The sampled (complete) dataset was then hidden for the remainder of computations. This procedure can be summarised as follows:

Task: Create three consistent datasets with variables $A_1, A_2, \ldots, A_n$.
A) Create a Bayesian network representing the data-generating distribution with density parameter d^*.
  A1) Insert d^* undirected edges between the variables uniformly at random.
  A2) Orientate edges according to enumeration to obtain DAG.
  A3) Set (conditional) probabilities uniformly at random.
B) Sample from this network to obtain dataset DS_0^1.
C) Create two further copies DS_0^2, DS_0^3.
D) Assign variables to datasets
  D1) A_1 measured by DS_1 and DS_2.
  D2) A_2 measured by DS_1 and DS_3.
  D3) A_3 measured by DS_2 and DS_3.
  D4) All other variables are assigned uniformly to the non-empty subsets of the power-set of $\{A_1, A_2, A_3\}$.
E) In DS_0^i delete all measurements of variables not measured by DS_0^i to give DS_i.
Output: DS_1, DS_2, DS_3.

Step 1 of OBN-cDS was carried out using the Matlab implementation of Tsamardinos et al. (2006).[7] No other source code was taken off-the-shelf. The triangulation (Step 3) was implemented

7. Implementation and testing on Matlab was chosen over R due to a number of helpful routines in Matlab's Causal Explorer Toolkit. The computational complexity of Step 1 has been investigated in, e.g., Ordyniak and Szeider (2013).

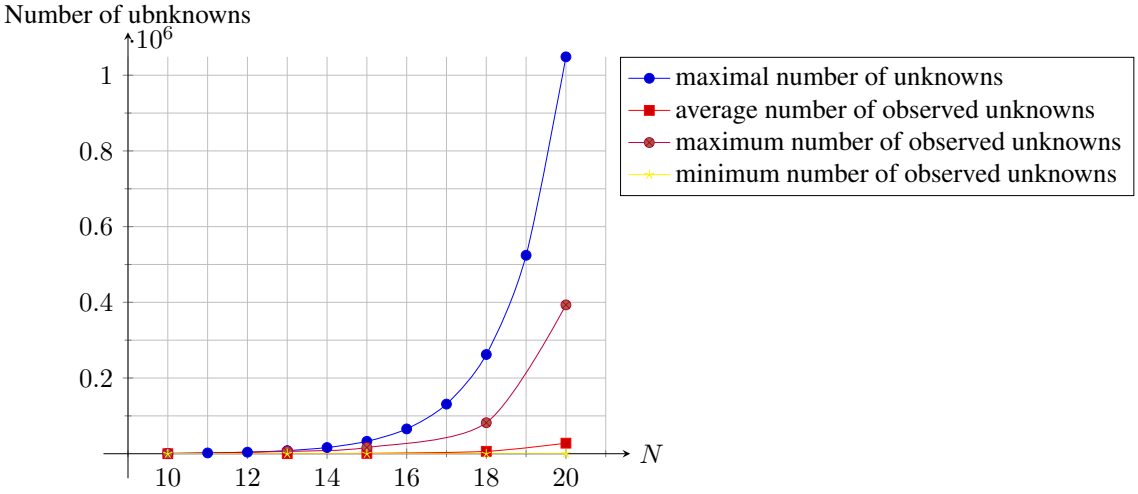


Figure 5: The number of unknowns needing to be determined by optimisation during Step 5b of OBN-cDS. N is the total number of variables. The chance network from which we sampled three datasets had constant $d^* \approx 21\%$ of the maximal number of arrows, $d^*_{\max} := \frac{N\cdot(N-1)}{2}$. The maximal number 2^N of unknowns is plotted in blue. Data points for $N = 10, 13, 15, 18, 20$ are all based on 200 iterations each. Plots are provided for the average observed number of unknowns in red, the maximum in purple and the minimum in yellow. The numerical values are reported in Table 2.

Table 2: The rows 2-5 report numbers of unknowns to be determined by optimisation, as depicted in Figure 5. The bottom row reports average run times in seconds to determine the number of unknowns for all 200 iterations in Figure 5. Where the number of unknowns is not zero, further computations are required to compute an OBN.

N	10	13	15	18	20
Average	53	288	930	6298	27927
Variance	60	581	1555	10140	41263
Maximum	380	6140	16380	81916	393212
Minimum	0	0	0	0	192
Run times	1.105	4.270	34.290	86.980	27,927.41

by writing a code for the simple P-Time triangulation algorithm presented by Berry (1999).[8] Implementation of the orientation (Step 4) was achieved by applying Williamson (2005a, Theorem 5.1) as explained in Section 3.3.1.

Step 5a was carried out by directly computing conditional probabilities from the datasets as explained in Section 3.3.3. Optimisation (Step 5b) was achieved by calling Matlab's optimisation routine 'fmincon'.

8. While there are more refined triangulation algorithms with improved worst-case complexity (Berry et al., 2006), we chose to implement the simpler version since in our application calculating triangulations only takes relatively little time compared to solving numerical optimisation problems.

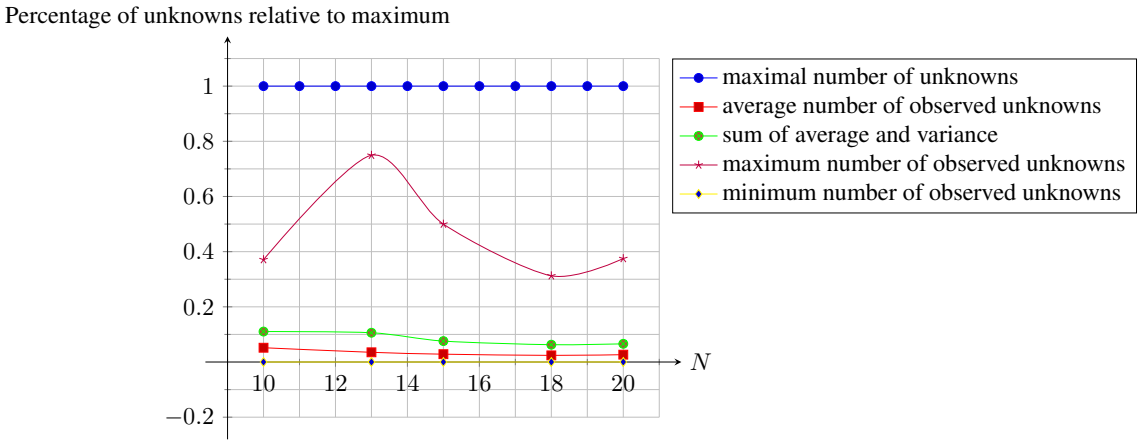


Figure 6: Replot of Figure 5 in relative rather than absolute terms concerning the unknowns (constant $d^* \approx 21\% d^*_{\max}$) for varying total number of variables N. The green curve displays the sum of the average and the variance.

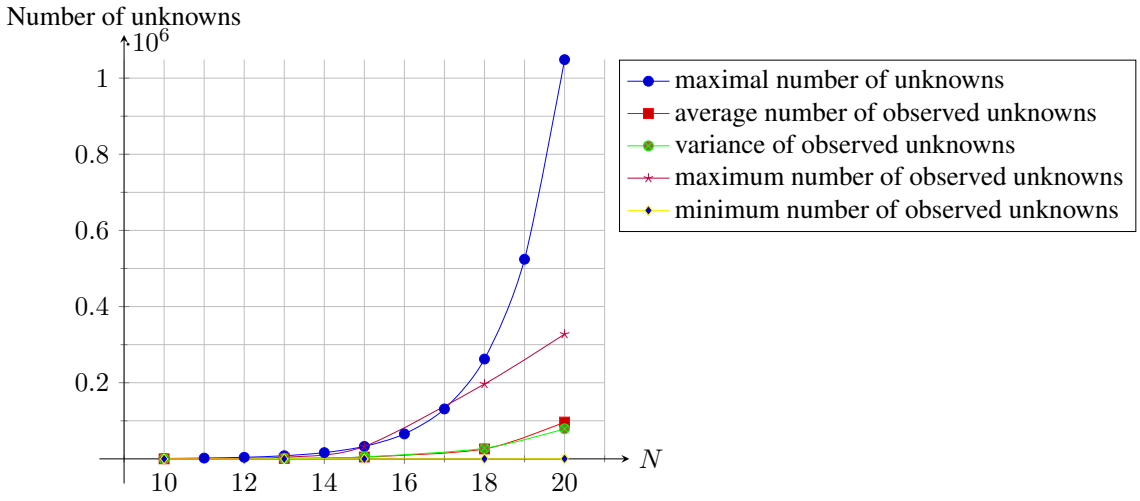


Figure 7: The number of unknowns needing to be determined by optimisation during Step 5b of OBN-cDS. N is the total number of variables. The chance network from which we sampled three datasets had constant $d^* \approx 31\%$ of the maximal number $d^* = \frac{N \cdot (N-1)}{2}$ of arrows. Compare with Figure 5, which uses $d^* \approx 21\%$. The maximal number of unknowns 2^N is shown in blue. Plotted in red is the Average, in green the variance, in purple the maximum and in yellow the minimum. The data point for $N = 20$ is based on 35 iterations. Data points for $N = 10, 13, 15, 18$ are based on 200 iterations each. The numerical values are reported in Table 3.

Table 3: The rows 2-5 report numbers of unknowns to be determined by optimisation in Figure 7. The bottom row reports average run times in seconds to determine the number of unknowns for all 200 iterations in Figure 5, with exception of the final column which is based on 35 iterations. Where the number of unknowns is not zero, further computations are required to compute an OBN.

N	10	13	15	18	20
Average	141	1230	4854	26032	96641
Variance	163	1021	5115	27899	79010
Maximum	1020	4108	32764	196604	327676
Minimum	0	0	0	120	384
Run times	0.965	56.575	36.170	117.290	17,265.543

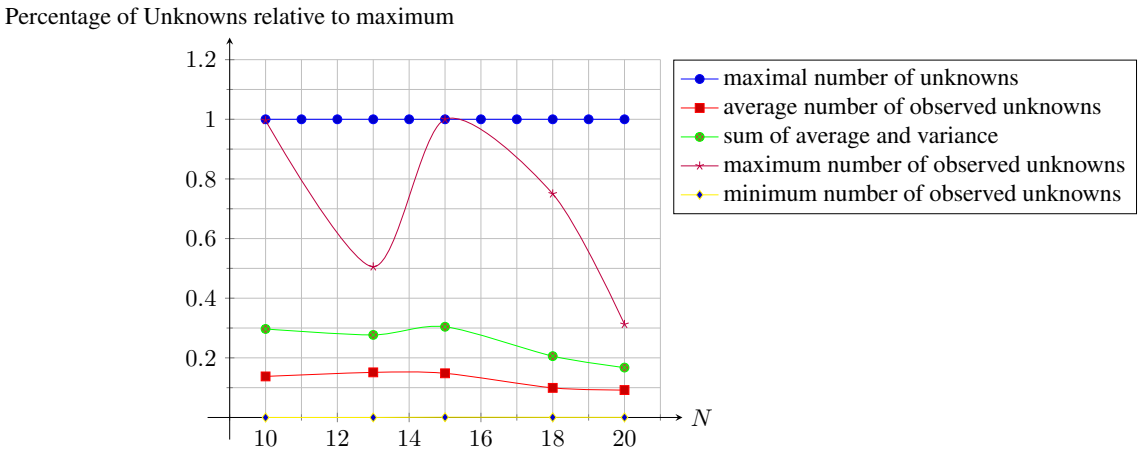


Figure 8: Replot of Figure 7 in relative rather than absolute terms concerning the unknowns (constant $d^* \approx 31\% d^*_{\max}$) for varying total number of variables N. The green curve displays the sum of the average and the variance.

Table 4: The first four rows report percentage of variables that are unknowns and to be determined by optimisation, as depicted in Figure 9. The bottom row reports average run times to determine the number of unknowns for all 200 iterations in Figure 9. Where the number of unknowns is not zero, further computations are required to compute an OBN.

d^*	10	14	20	25	30	35	40
Average	5.2	13.8	41.5	52.4	60.6	65.0	71.7
Variance	6.5	15.9	27.0	23.3	26.9	24.9	22.8
Maximum	37.1	99.6	99.6	99.6	99.6	99.6	99.6
Minimum	0	0	0	0	5.9	10.2	24.2
Run times	1.110	0.965	1.120	1.305	1.475	1.635	1.930

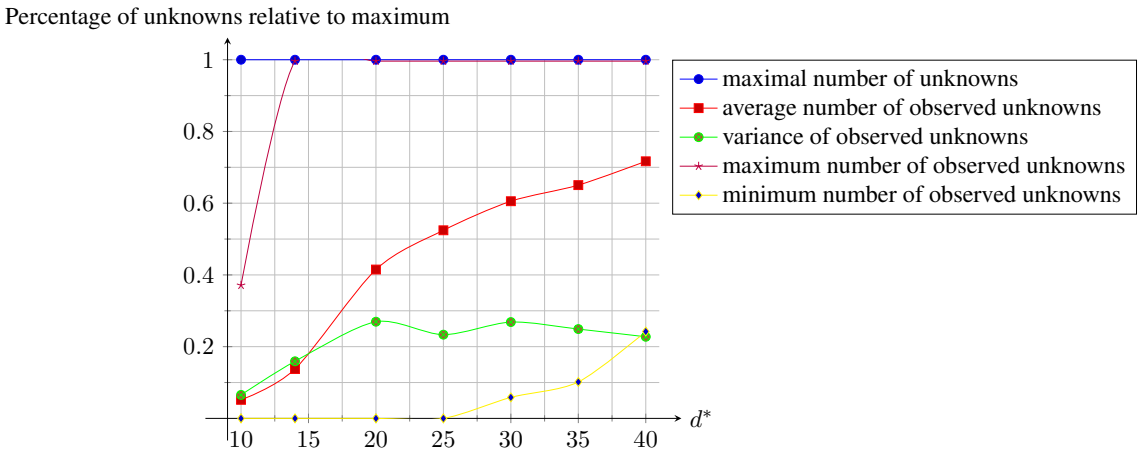


Figure 9: Here we kept $N = 10$ fixed and varied the density, d^*, of the chance net we used for sampling. $d^*_{\max} = 45$. We recorded the percentage of unknown variables that needed to be found by optimisation. Maximum number of unknowns of optimisation problem is $2^N = 1024$. We show data for $d^* \in \{10, 14, 20, 25, 30, 35, 40\}$. The graph depicts the maximal number of unknowns 2^N (in blue), the average (red), the variance (green), the maximum (purple) and the minimum (yellow). Every data point is based on 200 iterations. The numerical values are reported in Table 4.

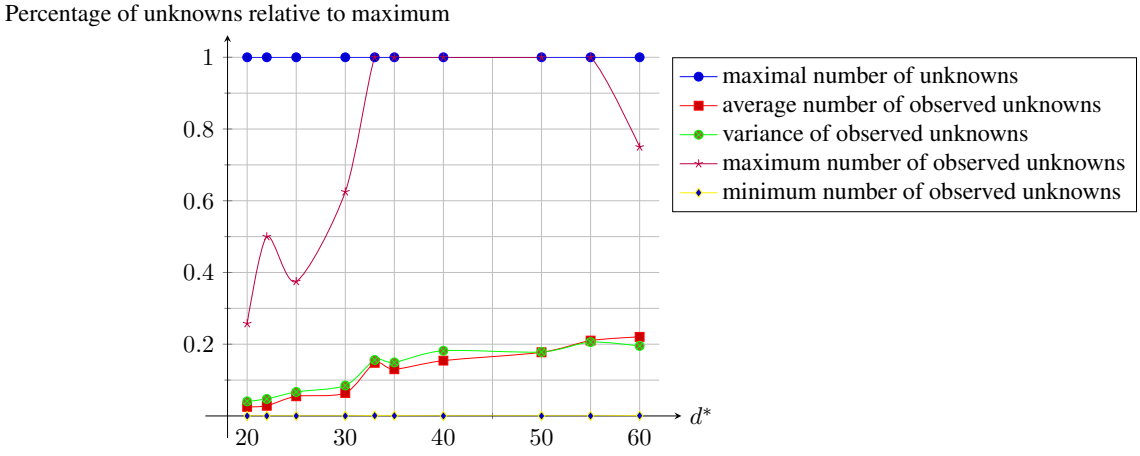


Figure 10: We kept $N = 15$ fixed and varied the density from the chance net, d^*, we used for sampling. Maximum number of unknowns of optimisation problem is $2^N = 32768$. We recorded the percentage of unknown variables in the OBN. Maximum number of unknowns of optimisation problem is $2^N = 1024$. We show data for $d \in \{20, 22, 25, 30, 33, 35, 40, 50, 55, 60\}$. The maximal number of unknowns 2^N is depicted in blue, the average in red, the variance in green, the maximum in purple and the minimum in yellow. Every data point is based on 200 iterations. The numerical values are reported in Table 5.

Table 5: The first four rows report the percentage of variables that are unknowns and to be determined by optimisation, as depicted in Figure 10. The bottom row reports average run times to determine the number of unknowns for all 200 iterations in Figure 10. Where the number of unknowns is not zero, further computations are required to compute an OBN.

d^*	20	22	25	30	33	35	40	50	55	60
Average	2.5	2.8	5.5	6.4	14.8	13.0	15.4	17.7	21.0	22.1
Variance	4.1	4.7	6.7	8.5	15.6	14.9	17.8	17.8	20.6	19.5
Maximum	25.7	50.0	37.5	62.5	100.0	100.0	100.0	100.0	100.0	75.0
Minimum	0	0	0	0	0.0	0.0	0	0	0	0.0
Run times	4.345	4.275	3.515	13.380	31.065	36.170	45.225	46.705	64.045	57.385

Table 6: The time taken to compute an OBN for $N = 15$ in 8 instances, presented in increasing order. Stated run times here include the time taken for solving the optimisation problem (Step 5b).

Number of Unknowns	0	18	36	36	48	192	228	408
Time taken in seconds	3	1.506E+3	8.72E+2	2.0840E+3	4.642E+3	9.896E+4	1.828E+6	2.561E+5

Table 7: Time taken to compute an OBN for $N = 10$ in 25 instances – put in increasing order. Rows 1 and 3 are the number of unknowns, rows 2 and 4 are the times taken in seconds. Stated run times here include the time taken for solving the optimisation problem (Step 5b).

0	0	0	0	0	0	0	0	0	0	0	0	8
4	5	6	6	6	7	7	7	7	7	8	11	14
8	8	16	16	24	28	36	60	68	172	188	220	
33	435	203	251	210	200	495	1266	9033	4.185E+4	3.312E+5	2.450E+5	

Table 8: Comparison between the time taken to compute an OBN (Lines 1 and 2) and the time taken to use the brute force approach (Line 3) for $N = 12$ and $d^* = 16$ in 9 instances.

Number of Unknowns	2	8	32	16	48	108	72	316	252
Time taken in seconds	5	246	454	673	991	8429	8629	78889	92138
Time taken in seconds	167	558	333	693	15403	194	362	341	529

Discussion. We used the freely available Matlab implementation of Tsamardinos et al. (2006) to learn Bayesian networks from data representing the P_i^*. Unfortunately, the code is encrypted as a pcode and cannot be modified without a password, which we were not able to obtain. This creates two issues: (i) newer versions of Matlab do not support pcode and (ii) compiling the pcode returns an error message and fails to output a Bayesian network, if there are too few probabilistic dependencies between measured variables. We addressed issue (i) by using an older version of Matlab. Issue (ii) is no real problem for our current application, since problem instances with few probabilistic dependence are the easy cases for OBN-cDS and where its superiority over the brute-force approach to entropy maximisation is apparent. We note that the issue only appeared for instances with twelve or less variables, $N \leq 12$. Furthermore, the issue became more frequent the smaller N and the lower the density of the chance network, d^*. Even for the smallest problems we tested, we saw the error message significantly less than one in ten instances. This means that the bias is washed out as the size N of the problem increases.

The concern of this section is to provide a proof of concept of OBN-cDS at an abstract level. The reliability of the employed structure learning algorithm (Step 1) is relevant but not of primary interest here, as such structure learning algorithms are already widely studied. The performance of MMHC is discussed by Aliferis et al. (2010); Gámez et al. (2010); Heinze-Deml et al. (2018); Nandy et al. (2018); Raskutti and Uhler (2018); Tsamardinos et al. (2006); Xie and Geng (2008). Unsurprisingly, the quality of the learned structure depends on the scoring function used to measure quality (Liu et al., 2012; Scutari et al., 2019). Moreover, it has been suggested that whether data sets used for learning were synthetic or from the real world influences the ranking of structure learning algorithms (Malone et al., 2015).

For every problem instance we created in which the pcode executed correctly, OBN-cDS returned a probability function. Matlab's in-built optimisation routine 'fmincon' applied to the maximisation of entropy over 2^N states (the brute-force approach) failed to produce an output for some of these instance because it ran out of memory and terminated calculations prematurely. This is not surprising, as for 13 variables the number of unknowns is equal to 8191.

In instances with few unknowns (under-determined variables), OBN-cDS, in general, outperformed the brute-force approach significantly in terms of computational speed (instances with few under-determined variables in Tables 3, 4 and 5 and as well as the first rows of Table 8). Numbers of under-determined variables are reported in Figures 5–10.

In a small number of instances, the brute-force approach found a probability function consistent with the constraints (within the specified optimisation parameters) which also had greater entropy than the function output by the implementation of OBN-cDS, in a shorter time. Our best explanation is that the brute-force approach poses a convex optimisation problem (convex objective function and linear constraints) while our method poses a potentially non-convex optimisation problem (non-convex objective function and multi-linear constraints). There are hence, in general, multiple local optima in our method and the optimisation algorithm outputs a local rather than a global optimum.

In sum, OBN-cDS is superior to the brute-force approach where there are few under-determined variables or where the number of variables becomes so large that the brute-force approach is no longer feasible. The speed-up strategies developed in the remainder of this paper are likely to lead to further efficiency savings, although these were not tested here. Our overall recommendation is to apply the brute-force approach where computationally feasible and switch to our algorithm once the brute-force approach is no longer feasible.

Our approach has the added benefit that it determines many of the parameters of the network without needing to solve an optimisation problem. For applications which do not require specifying a probability function over the entire domain, further efficiency savings are possible by parameterising just the part of the OBN that is required in practice.

4. Computational Considerations

We now consider general computational questions concerning the general algorithm OBN-cDS. In Sections 4.1–4.4, we discuss ways to speed up the algorithm in special cases. Worst-case complexity is discussed in Section 4.5.

4.1 General Considerations

OBN-cDS allows a number of free choices. In order to reduce the size of the optimisation problem, the following properties are desirable (see Figure 11 for an example):

1. As few under-determined variables as possible.
2. Under-determined variables with lower arity, if possible.
3. As few fill-in edges as possible.
4. A cut vertex[9] of $\mathcal{G}^T$ with maximal degree[10] as root.
5. Under-determined variables as close as possible to the root.
6. As few as possible under-determined variables that have an under-determined variable as ancestor/descendant.

These properties are desirable because:

1. The fewer under-determined variables there are, the fewer the number of unknowns in (1); see Section 3.3.3 and Section 4.2.
2. The lower the arity of the under-determined variables, the fewer the number of unknowns in (1).
3. Sparser graphs are likely to have fewer under-determined vertices, with fewer parents; see Section 3.3.3. Also, the sparser the graph, the faster one can compute the objective function (1).
4. A cut vertex of $\mathcal{G}^T$ as a root vertex entails that maximising (1) can be achieved by independently maximising entropy in the remaining connected components; see Section 4.4 and Figure 11.
5. The closer the under-determined vertices are to the root, the fewer terms are in the constraints; see Section 3.3.4.

9. A cut vertex is a node such that, if one removes this node and all arrows that point to or point from this node, then the number of connected components increases. Intuitively, removing this node and severing all connections with this node cuts the graph into more pieces.
10. The *degree* of a vertex is the number of edges incident on this vertex.

6. The more separate pathways the under-determined vertices are on, the more the optimisation problem can be split into *independent* optimisation problems; see Theorem 33.

Furthermore, if a y-parameter y and all its parents are fully determined, then all summands in the objective function which contain the term $\log y$ consist only of known expressions. Hence, these summands are fixed (irrelevant) for maximising entropy and the objective function can be pruned of these terms.

Note that it may not be possible for these desiderata to all be attained at the same time. As a rule of thumb, we would suggest minimising the number of under-determined variables first, as this reduces the dimensionality of the optimisation problem.

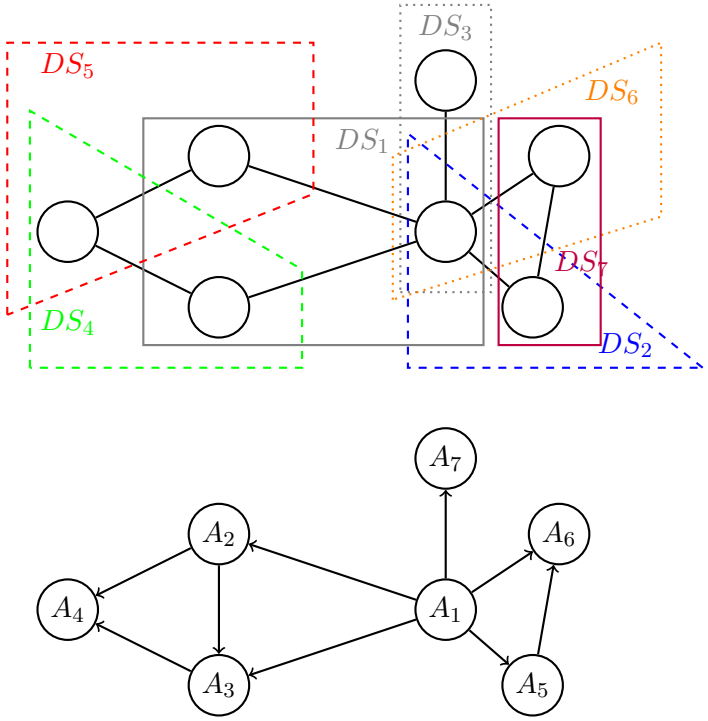


Figure 11: Top: Undirected graph $\mathcal{G}$ on 7 variables measured by seven datasets. Bottom: Clever choices of an enumeration of variables and orientation of arrows produces the graph of an OBN with only two under-determined variables (A_4 and A_6) which are both relatively close to the root. Also note that both under-determined variables only have two parent variables. Furthermore, the optimisation problem decomposes into two independent problems, since A_1 is a cut vertex.

We should note that in order to address (3), there is one way of avoiding fill-in edges altogether: instead of using the algorithm to determine a Bayesian net, use it to determine a Markov net, which represents the maximum entropy function by means of an undirected graph, rather than a DAG. This would eliminate steps (3) and (4) of OBN-cDS and would require modifying (5). We have not taken this approach in the present paper because Bayesian nets (and triangulated Markov nets) are generally regarded as more convenient for probabilistic inference than Markov nets, but we note this option here as one that offers benefits in terms of the computational complexity of constructing the net.

We next show how to jointly address Properties 1–3.

4.2 Healthy Minimal Triangulations

Ideally, one would like to find a set of fill-in edges (Step 3 of OBN-cDS) which triangulates the overarching Markov net structure $\mathcal{G}$ and which is contained in every other triangulation. Unfortunately, no such canonical fill-in exists, as explained in Figure 2. The next best option would be to compute a minimum triangulation, a triangulation with the minimum number of fill-in edges. However, the problem of computing a minimum triangulation is NP-complete.[11] While computing minimum triangulations may be computationally infeasible, computing minimal triangulations is not. Polynomial-time minimal triangulation algorithms are surveyed by Heggernes (2006).

Whether or not a variable is under-determined depends on the orientation of arrows which, in turn, depends on the enumeration. A fill-in edge between variables which have not been jointly measured is undesirable since this *inevitably* results in at least one under-determined vertex in the final Bayesian net—regardless of the orientation of the edges. We now show how to avoid one type of these troublesome fill-in edges, which we call an 'unhealthy' edge, by computing a particular minimal triangulation in polynomial time.

Definition 12. The *appendix* A^i of dataset DS_i in a collection of datasets consists of those variables measured by dataset DS_i which are not measured in any other dataset. An *unhealthy* edge is an edge which connects two appendices. A graph on V is called *healthy*, if and only if it does not contain an unhealthy edge. Graphs containing an unhealthy edge are said to have *appendicitis*.

Definition 13. The *neighbourhood* of a set of vertices S in an undirected graph, $N(S)$, is defined as the set of vertices in the complement of S which are connected by an edge to S.

The neighbourhood of an appendix A^i, $N(A^i)$, consists of those vertices connected by an edge to a vertex in A^i. In $\mathcal{G}$ (Step 3), we trivially have that $N(A^i) \subseteq DS_i$.

We now give the first part of an algorithm which, as we show below, quickly computes a minimal and healthy triangulation of $\mathcal{G}$:

Input: Undirected graph $\mathcal{G}$ computed in Step 3.
I) For all i and all $A, B \in N(A^i)$: compute a fill-in edge between A and B, if and only the edge between A and B is a chord of somesimple cycle c in $\mathcal{G}$ of length four or greater.
II) Add all these fill-in edges to $\mathcal{G}$ yielding graph $\mathcal{G}^-$.
III) For all i: drop all vertices not in $A^i \cup N(A^i)$ from $\mathcal{G}^-$ and compute a minimal triangulation.
IV) Drop all vertices contained in some appendix from $\mathcal{G}^-$ and compute a minimal triangulation.
V) Add the fill-in edges computed in Step III and Step IV to $\mathcal{G}^-$ yielding graph $\mathcal{G}^+$.
Output: Undirected graph $\mathcal{G}^+$.

The total number of runs of a triangulation algorithm is $2h + 1$ (where h is the number of available datasets). So, the more datasets there are, the more time the algorithm takes. In most applications, when considering large optimisation problems it is the number of variables and/or the number of observations which quickly grow. While the number of datasets h may also grow, we believe that it is sensible to assume that h *grows at most in a linear rate in terms of the input size*

11. Computing a minimum triangulation is NP-complete in terms of the input *graph*. In applications with large datasets and a moderate number of variables, it might be preferable to invest the extra time to compute a minimum triangulation and subsequently obtain a sparser net with fewer under-determined variables.

(size of datasets). Computing a minimal triangulation is achievable in polynomial time in terms of input graphs (Heggernes, 2006).

Definition 14 (Separator)**.** A minimal separator of $\mathcal{G}$ is a vertex set $S \subset V$ such that $V \setminus S$ has at least two components C_1, C_2 such that $N(C_1) = N(C_2) = S$. A separator of $\mathcal{G}$ is a vertex set which contains a minimal separator.

Note that for non-empty A^i the sets $N(A^i)$ and $N(A^i) \cap N(V \setminus (A^i \cup N(A^i)))$ are both separators of $\mathcal{G}$—not necessarily minimal.

Intuitively, the chords computed in Step 1 cut every simple cycle which connects an appendix with vertices which are not in the neighbourhood of this appendix. One can think of $\mathcal{G}^-$ as the result of creating separators in the $N(A^i)$ which shield the appendices from vertices not in their neighbourhood.

Proposition 15. *$\mathcal{G}^+$ is healthy and $\mathcal{G}^+$ is a triangulation of $\mathcal{G}$ which, on input $\mathcal{G}$, can be computed in polynomial time.*

See Appendix B for the proof, as well as for the proof of Proposition 16.

We now give the second part of our polynomial-time algorithm to compute a triangulation of $\mathcal{G}$ which is healthy and minimal.

Input: Undirected graph $\mathcal{G}^+$.
VI) Drop all edges added in Step II from $\mathcal{G}^+$ yielding $\mathcal{G}_0$.
VII) Run a polynomial time minimal triangulation algorithm on $\mathcal{G}_0$ but only add edges which connect two vertices in $N(A^i)$.
Output: $\mathcal{G}^{++}$.

Proposition 16. *$\mathcal{G}^{++}$ is a healthy and minimal triangulation of $\mathcal{G}$ which can be computed in polynomial time.*

4.3 Resolving Appendices

The number of under-determined variables directly influences the number of unknowns to be computed by optimisation. In order to keep the number of under-determined variables low, it is preferable to avoid arrows in the OBN which originate from one appendix and end in a different appendix. For every such arrow, the variable the arrow points to is under-determined. Furthermore, arrows pointing from a variable in appendix $A_i \in A^i$ towards a variable A_l measured in the same dataset but not in the appendix, $A_l \in DS_i \setminus A^i$, potentially give rise to an under-determined variable. We now show how to find (the DAG of) an OBN, such that all edges originating in an appendix also point towards a variable in that same appendix.

Proposition 17 (No Appendicitis)**.** *We can find an OBN in which all arrows beginning in an appendix also end in this same appendix. Furthermore, every variable in an appendix is fully determined.*

In particular, these conditional probabilities in the appendices can be found without having to solve either an optimisation problem or an algebraic problem: they can be obtained directly from

the data. Obtaining conditional probabilities in the appendices comes at the cost of adding further edges to $\mathcal{G}$.

So, under-determined variables can only be found outside of appendices. Here is the pseudo code.

Input: consistent datasets $DS_1, \ldots, DS_h$.
1) For all datasets DS_i learn a Markov network structure $\mathcal{G}_i$ from DS_i representing independences of P_i^*.
2) Set overarching undirected graph $\mathcal{G}$ as the union of the $\mathcal{G}_i$.
3) For all datasets DS_i and all pairs of distinct variables A_l, A_k measured by DS_i that are not in the appendix A^i add an undirected edge between A_i and A_l.
4) Compute a minimal healthy triangulation of this graph.
5) Pick an enumeration of the variables in Step 4 of OBN-cDS such that for all datasets DS_i and all variables in its appendix $A_i \in A^i$ are enumerated after all variables not in the appendix A^i.
6) Define an orientation of the edges as in OBN-cDS.
Output: DAG of an OBN that is healthy.

The idea is to enumerate variables in appendices only after all other variables in the dataset have been enumerated.

In the worst case, there is no variable which is only measured in a single dataset. Hence, there are no appendices and the above does, in general, not result in any computational speed up. However, the more variables there are which are only measured in a single dataset (the larger the appendices), the more promising the above transformation appears to be in terms of a computational speed up.

4.4 Connection Lost—Cut Vertex as Root

The computational hardness of optimisation problems generally increases super-linearly in the number of unknowns to be determined. It is hence preferable to decompose optimisation problems into multiple independent problems. We now show that this is possible here for a class of problems.

If $\mathcal{G}^T$ has more than one connected component consisting, respectively, of the vertices $\{A_i | i \in I_1\}, \{A_i | i \in I_2\}, \ldots, \{A_i | i \in I_c\}$ then the objective function of the optimisation problem can be decomposed as follows:

$$-\sum_{r=1}^{c} \sum_{i_r \in I_r} \sum_{v_r @ Anc'_{i_r}} \Big(\prod_{A_j \in Anc'_{i_r}} y_j^{v_r} \Big) \log y_{i_r}^{v_r}. \tag{8}$$

The two inner sums only contain terms pertaining to one single connected component, I_r.

Furthermore, the constraints can also be decomposed. We can hence find an OBN by independently maximising entropy on every connected component, that is we solve for non-negative y-parameters, for all $1 \leq r \leq c$

$$\begin{aligned} \text{maximise: } & -\sum_{i_r \in I_r} \sum_{v @ Anc'_{i_r}} \Big(\prod_{A_j \in Anc'_{i_r}} y_j^{v_r} \Big) \log y_{i_r}^{v_r} \\ \text{subject to: } & \sum_{g @ \{A_i\}} y_i^{g v_r} = 1 \text{ for all } i \in \mathcal{U} \cap I_r, \end{aligned}$$

$$\sum_{s@Anc_i \setminus Par_{i:k}} \prod_{A_j \in Anc'_i} y_j^{sv_r} = P_k^*(a_i^v Par_{i:k}^{v_r}) \text{ for all } i \in \mathcal{U} \cap I_r \text{ and}$$

for all maximal jointly measured sets of the form $\{A_i\} \cup Par_{i:k}$.

By choosing a cut vertex of $\mathcal{G}^T$ as root of the OBN with the connected graph $\mathcal{H}$ we can exploit this. Let A_o be the root and cut vertex. Let $J_1, \ldots, J_c$ be the connected components which remain after removing A_o. Then the objective function equals:

$$-\sum_{r=1}^{c} \sum_{i_r \in J_r} \sum_{v@Anc'_{i_r}} y_o^v \cdot \Big(\prod_{A_j \in Anc'_{i_r}} y_j^{v_r} \Big) \log y_{i_r}^{v_r}. \tag{9}$$

Since the y_o^v can be obtained via Step 5a we have decomposed the objective function in a way similar to that of (8). Clearly, the constraints decompose again, too. We can hence find an OBN by independently solving for c *independent* optimisation problems.

Example 18. Suppose there exists a dataset DS_i which only measures variables that are not measured by any other dataset DS_j. This means that all variables measured by DS_i are in the appendix A^i. An OBN is then obtained by (i) determining an OBN for the variables measured by all other datasets DS_j, (ii) learning an OBN for DS_i directly from the data and (iii) taking the union of these two OBNs.

4.5 Complexity of OBN-cDS

In typical cases, $\mathcal{G}^T$ is a sparse graph, and maximising (1) is computationally much simpler than brute-force maximisation of entropy $H(P) = -\sum_{v@V} P(v) \log P(v)$ expressed in terms of exponentially many states of V (Williamson, 2005a, p. 95). Roughly speaking, the sparser the graph, the fewer conditional dependencies there are, and the fewer the y-parameters there are in (1). This leads to an optimisation problem with fewer unknowns. In concrete applications, further simplifications may be achievable by following the recipe given in Section 4.1.

We shall now discuss the computational complexity of the general algorithm OBN-cDS.

4.5.1 Steps 1–5a

Although maximum likelihood learning of Markov nets is NP-complete (Srebro, 2003), this is largely because parameter learning is a complex task (Koller and Friedman, 2009, p.950). As Bromberg et al. (2009) argue, structure learning can be performed relatively efficiently. This remains the case if Markov net structure learning proceeds via Bayesian net structure learning, as in our proof of concept, discussed in §3.4. The complexity of learning Bayesian networks representing marginal frequencies in Tsamardinos et al. (2006) is polynomial, as long as the maximal degree of a vertex is bounded by a polynomial. This algorithm has been implemented in R and is freely available (Scutari, 2010) and it has also been implemented in Matlab's Causal Explorer Toolkit (Aliferis et al., 2003).

One can find a minimal triangulation in polynomial time (Berry et al., 2004). Maximum cardinality search can be completed in linear time (Berry et al., 2004, §3). Since $\mathcal{G}^T$ is triangulated, it has at most $|V|$ cliques Fulkerson and Gross (1965) which can be found in linear time (Berry and Pogorelcnik, 2011). Orienting all arrows is achievable in polynomial time (Dor and Tarsi, 1992). Steps 2–5 are hence computable in polynomial time.

Step 5a: Determining the conditional probability of a determined variable requires linear time in the product of i) the number of observations in dataset DS_i and ii) the number of variables measured in dataset DS_i: for every observation in the dataset DS_i determine the value of the variable and the values of the parents. Finally, divide the number of such observations by the total number of observations in DS_i.

4.5.2 STEP 5B

Denoting by $|A^i|$ the number of variables in appendix A^i and by h the number of datasets, we can give an upper bound to the number edges in $\mathcal{G}^{++}$:

Proposition 19. *The number of edges in $\mathcal{G}^{++}$ is at most*

$$\binom{n}{2} - \sum_{i=1}^{h-1} \sum_{j=i+1}^{h} |A^i| \cdot |A^j|.$$

Proof: There are n variables in V. The complete graph on n vertices has $\binom{n}{2}$ edges. Since $\mathcal{G}^{++}$ does not have appendicitis, no edge connects different appendices. Hence, there is no edge connecting any pair of variables in different appendices. For every fixed pair of different appendices A^i, A^j there are $|A^i| \cdot |A^j|$ pairs of variables. ■

A more natural measure of complexity than the number of edges is perhaps the number of y-parameters, which we shall now investigate.

For a fixed OBN let Cl_{max} be the maximal number such that there exists a dataset DS_i which measures Cl_{max} variables which are not in the appendix A^i such that there is an edge between every two of these variables. Denote such a set by Cl. So, the graph of the OBN contains a complete graph of size Cl_{max} as a sub-graph and all variables in this complete graph are jointly measured by DS_i and none of these variables is only measured by DS_i. The smaller the number of variables that are measured in appendices, the greater our prospects of being in a situation in which Cl_{max} is large.

Corollary 20. *If all variables are binary, then number of y-parameters which cannot be determined by Step 5a is in the worst case*

$$2^{n-\sum_{i=1}^{h} |A^i|} - 2^{Cl_{max}}. \tag{10}$$

Proof: In the worst case, all variables not in an appendix are connected to each other, which translates into $2^{n-\sum_{i=1}^{h} |A^i|}$ possible assignments of these variables.

We can start our maximum cardinality search algorithm at a variable in Cl and next visit all the other members of Cl. This means that we can determine the conditional probabilities at the first Cl_{max} variables in this enumeration by Step 5a. There are $2^{Cl_{max}}$ fully determined such y-parameters. ■

The brute-force approach to maximising entropy leads to an optimisation problem with $2^n - 1$ unknowns. Ceteris paribus, the more appendices there are and the larger they are, the greater is the reduction to the number of unknown parameters of OBN-cDS in the worst-case when compared to the brute-force method. We also want to point out that for the minimal sensible h, $h = 2$, we will show in Section 5 that we can compute *all* y-parameters via Step 5a, if both appendices are

non-empty. If both appendices are empty (i.e., there does not exist an appendix), then both datasets measure the exact same set of variables and determining an OBN becomes trivial. If one appendix is empty, then the other dataset measures all variables in V and determining an OBN becomes trivial.

As soon as the number of datasets is greater or equal than 3 ($h \geq 3$), Step 5a does not, in general, suffice to completely determine an OBN; see Section 7.2. It may be the case that there are no appendices. In such a case ($\sum_{i=1}^{h} |A^i| = 0$ and Cl_{max} small), the worst-case complexity (in terms of the number of unknowns to be determined) of our general approach is not much lower than that of the brute-force approach.

4.5.3 SUMMARY

As long as the maximal degree of a vertex is bounded by a polynomial, the complexity of learning the initial Markov net structures is polynomial (Step 1). Steps 2–4 can be computed in P-Time in terms of *input graphs*. Every application of Step 5a runs in linear time, if arities of variables and the number of parents is bounded by some constant. Step 5b requires the solution of an optimisation problem, which has fewer variables than the brute-force maximum entropy optimisation problem (Propositions 17 and 19 and Corollary 20). For these reasons, the computational complexity of OBN-cDS compares favourably with that of the brute-force method (see Section 3.4).

In subsequent sections, we shall on occasion add auxiliary edges to $\mathcal{G}$; this does not invalidate any of the relevant formal properties. At worst, adding further edges may lead to a slightly more complex optimisation problem. The remainder of this paper is devoted to finding classes of cases in which—with help of variations of the general algorithm OBN-cDS—we can find an OBN particularly quickly.

5. Two Datasets

The simplest case in which data integration becomes non-trivial is the case of two datasets. We explore this case in detail in this section, showing that an OBN can be obtained directly from the input datasets without needing to solve any optimisation problem.

5.1 Problem Description

If the set of variables of one dataset contains those of the other, then trivially, because the datasets are assumed to be consistent, the maximum entropy function is fully determined by the measured frequencies of the larger dataset. If the intersection of variables measured in the two datasets is empty, then the maximum entropy function over V is simply the product of the measured frequencies (Example 18). We thus consider two datasets where neither set of measured variables is contained in the other set of measured variables and the intersection of measured variables is non-empty.

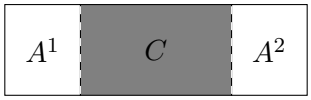
We consider two datasets DS_1 and DS_2, where DS_1 measures variables $A^1_1, \ldots, A^1_{l_1}, C_1, \ldots, C_k$ and DS_2 measures the variables $A^2_1, \ldots, A^2_{l_2}, C_1, \ldots, C_k$ with $k, l_1, l_2 \geq 1$ and $A^1 := \{A^1_1, \ldots, A^1_{l_1}\}$, $A^2 := \{A^2_1, \ldots, A^2_{l_1}\}$ disjoint. We refer to $C := \{C_1, \ldots, C_k\}$ as the *centre* and to A^1, A^2 as *appendices*. An example can be found in Table 9, and a more abstract depiction is presented in Figure 12.

We next show how to obtain an OBN directly from the datasets, without solving any optimisation problem.

Table 9: Example of two consistent datasets ($P_1^*(survived) = \frac{2}{3} = P_2^*(survived)$) which between them measure set of variables $V = \{\text{Sex}, \text{Treatment Outcome}, \text{Age}\}$. The black cells indicate observations not made.

		Variables measured in Dataset 1		
			Variables measured in Dataset 2	
	Observation	Sex	Treatment Outcome	Age
Observations of Dataset 1	Patient 1	Male	survived	
	Patient 2	Female	died	
	Patient 3	Female	survived	
Observations of Dataset 2	Patient 4		survived	$60 +$ years
	Patient 5		survived	$60 +$ years
	Patient 6		died	$40 - 59$ years
		Variables measured in Appendix 1	Variables measured in the Centre	Variables measured in Appendix 2

Figure 12: Schematic representation of variables for two datasets with non-trivial intersection of measured variables. The healthy graphs are these graphs in which there is no edge connecting A^1 and A^2.



5.2 OBN-2cDS

We find an OBN from two consistent datasets via the following algorithm, which we call OBN-2cDS:

Input: datasets DS_1, DS_2.
1) Run OBN-cDS on centre C to determine OBN $\mathcal{B}_C$ for C.
2) For i=1,2 learn a Markov network structure $\mathcal{F}_i$ from DS_i.
3) For i=1,2 add edges to $\mathcal{F}_i$ to ensure that the centre nodes form a clique, yielding $\mathcal{G}_i$.
4) For i=1,2 compute a minimal triangulation $\mathcal{G}_i^T$ of $\mathcal{G}_i$.
5) For i=1,2 run maximum cardinality search on $\mathcal{G}_i^T$, ensuring that the centre nodes are enumerated before nodes in appendix A^i and have the same ordering for all i.
6) For i=1,2 orientate $\mathcal{G}_i^T$ to give DAG $\mathcal{H}_i$.
7) For i=1,2 determine conditional probabilities of all variables in A^i given their parents in $\mathcal{H}_i$ directly from DS_i.
8) Obtain overarching DAG $\mathcal{H}$ by taking the union of the $\mathcal{H}_i$ and then replacing the arrows on the centre nodes by those in $\mathcal{B}_C$.
9) Take as parameters those in $\mathcal{B}_C$ as well as those obtained in (7).

Output: Objective Bayesian Net

The correctness of OBN-2cDS is proved below. By inspecting the algorithm, the OBN is obtained directly from the DS_i. No optimisation problem needs to be solved.

In contrast, running OBN-cDS directly on the two datasets will, in general, require the solution of an optimisation problem. This is because OBN-cDS can place a vertex in the centre with parents in both appendices. This variable would be under-determined and conditional probabilities would have to be found via optimisation.

5.3 Analytic Representation of the Maximum Entropy Function

We next determine the analytic representation of the maximum entropy function, which we shall later make use of.

First note that consistency implies that for all $v_0@C$, $P_1^*(v_0) = P_2^*(v_0)$. We hence drop the index and simply write $P^*(v_0)$.

Two non-empty disjoint sets of variables $X, Y \subset V$ are conditionally independent given all other variables, according to $P^\dagger$, if no variable in X occurs in a dataset which mentions a variable in Y (see Section 3.2). To see this, first define an undirected graph with the variables in V as nodes and an edge between each pair of variables that are jointly measured by some dataset. If a subset of variables $S \subset V$ separates X from Y in this graph, then X and Y are conditionally independent given S, according to $P^\dagger$; see Williamson (2005a, Theorem 5.6).

We hence find that for all assignments $v_0@C, v_1@A^1, v_2@A^2$ such that $P^\dagger(v_0v_1v_2)$ is non-zero,

$$\begin{aligned} P^\dagger(v_0v_1v_2) &= P^\dagger(v_1v_2|v_0) \cdot P^\dagger(v_0) \\ &= P^\dagger(v_1|v_0) \cdot P^\dagger(v_2|v_0) \cdot P^\dagger(v_0) \\ &= P_1^*(v_1|v_0) \cdot P_2^*(v_2|v_0) \cdot P^*(v_0) \ . \end{aligned} \tag{11}$$

$P^\dagger(v_0v_1v_2)$ is zero if and only if $P_1^*(v_0v_1) = 0$ or $P_1^*(v_0v_2) = 0$, by the open-mindedness property of $P^\dagger$ (Paris, 1994, p. 95). Since (11) holds for all such assignments, these conditions fully determine the maximum entropy function $P^\dagger$.

5.4 Correctness of OBN-2cDS

We note that Step 1 of OBN-2cDS determines an OBN for $P^\dagger(v_0) = P^*(v_0)$; since the output of the algorithm agrees with Step 1 in the centre, the algorithm correctly outputs a DAG and conditional probabilities in the centre.

To establish correctness of OBN-2cDS, we now turn to the variables in the appendices.

Note that $\mathcal{H}_i$ is the DAG of an OBN representing P_i^*. This follows since we constructed $\mathcal{H}_i$ by the same procedure as that used by OBN-cDS.

We are interested in determining the conditional probabilities $P_i^*(v_i|v_0)$ for all $v_i@A^i, v_0@C$. These conditional probabilities have to satisfy the independence relations represented by $\mathcal{H}_i$, but they might satisfy further independence relations. The DAG $\mathcal{H}$ of an OBN for $P^\dagger$ is thus obtained by taking as vertices $C \cup A^1 \cup A^2$ and as arrows all the arrows of $\mathcal{B}_C$ and those arrows in $\mathcal{H}_1$ and $\mathcal{H}_2$ which do not connect two vertices in C.

We now show that $\mathcal{H}$ is acyclic. There cannot be any cycle in the centre, since all directed edges in the centre are those of the DAG $\mathcal{B}_C$. Similarly, there cannot be any cycle in an appendix, since

all directed edges in an appendix are those of a DAG. If there were a cycle containing a variable in the centre and a variable in an appendix, then there would have to be an arrow originating from an appendix and pointing to the centre. By construction, there is no such arrow. Hence, $\mathcal{H}$ is acyclic.

All conditional probabilities in the centre can be read-off from $\mathcal{B}_C$. The conditional probabilities in the two appendices are obtained as explained in Section 3.3.3. So:

Proposition 21. *In the case of two consistent datasets, OBN-2cDS finds an OBN without underdetermined variables.*

In sum, then, in the two-dataset case, we obtain an OBN directly from the datasets. There is no need to maximise entropy by solving an optimisation problem (1).

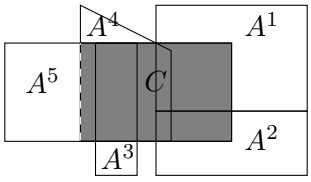
6. Centred Datasets

In this section, we show how the results for two consistent datasets can be generalised to larger collections of datasets. We show that if the datasets are *centred* then an OBN can be found without needing to solve an optimisation problem.

Definition 22. A collection of $h \geq 2$ datasets is *centred*, if and only if there exists a dataset DS_m such that every variable which is measured in more than one dataset is also measured in DS_m. The variables in DS_m are $A^m_1, \ldots, A^m_{l_m}$ and $C_1, \ldots, C_k$. The variables in dataset DS_i with $i \neq m$ are the $A^i_1, \ldots, A^i_{l_i}$, which are unique to DS_i, together with some variables from $C_1, \ldots, C_k$. All variables are distinct. We extend the notions of the centre C and an appendix A^i in the obvious way.

In particular, every collection of two datasets is centred; the centre consists of the variables measured by both datasets. An example of a collection of centred datasets can be found in Figure 13.

Figure 13: DS_5 measures all variables in the centre; indicated by the dashed line. All other datasets measure the variables in their respective appendix and a subset of the variables in the centre. Every variable measured in two or more datasets is contained in the centre.



Again, we can obtain an OBN directly from the datasets, without solving any optimisation problem. The underlying idea is simple: repeat the argument for two datasets but replace the number 2 by the number of datasets, h.

6.1 OBN-ccDS

For consistent centred datasets, we can find an OBN via the following algorithm, which we call OBN-ccDS:

Input: centred datasets $DS_1, DS_2, \ldots, DS_h$.
1) Run OBN-cDS on centre C and determine OBN $\mathcal{B}_C$ for C.
2) For $i = 1, 2, ..., h$ learn a Markov net structure $\mathcal{F}_i$ from DS_i.
3) For $i = 1, 2, ..., h$ add edges to $\mathcal{F}_i$ to ensure that the centre nodes form a clique, yielding $\mathcal{G}_i$.
4) For $i = 1, 2, ..., h$ compute a minimal triangulation $\mathcal{G}_i^T$ of $\mathcal{G}_i$.
5) For $i = 1, 2, ..., h$ run maximum cardinality search on $\mathcal{G}_i^T$, ensuring that the centre nodes are enumerated beforenodes in appendix A^i and have the same ordering for all i.
6) For $i = 1, 2, ..., h$ orientate $\mathcal{G}_i^T$ to give DAG $\mathcal{H}_i$.
7) For $i = 1, 2, ..., h$ determine conditional probabilities of all variables in A^i given their parents in $\mathcal{H}_i$ from DS_i.
8) Obtain overarching DAG $\mathcal{H}$ by taking the union of the $\mathcal{H}_i$ and then replacing the arrows on the centre nodes by those in $\mathcal{B}_C$.
9) Take as parameters those in $\mathcal{B}_C$ as well as those obtained in (7).
Output: Objective Bayesian Net

6.2 Analytic Representation of the Maximum Entropy Function

The analytic solution is analogous to the two dataset case. Again, the variables measured in DS_i which are in the centre C screen off appendix A^i from all other variables. This follows exactly as above. Denote by v_0^i the restriction of an assignment $v_0@C$ to the variables measured in DS_i. For assignments all $v_0@C, v_i@A^i$ such that $P^\dagger(v_0 v_1 \ldots v_N)$ is non-zero we find

$$\begin{aligned} P^\dagger(v_0 v_1 \ldots v_N) &= P^\dagger(v_1 \ldots v_N | v_0) \cdot P^\dagger(v_0) \\ &= P^\dagger(v_0) \cdot \prod_{i=1}^{N} P^\dagger(v_i | v_0) \\ &= P^\dagger(v_0) \cdot \prod_{i=1}^{N} P^\dagger(v_i | v_0^i) \\ &= P_m^*(v_0) \cdot \prod_{i=1}^{N} P_i^*(v_i | v_0^i). \end{aligned} \tag{12}$$

$P^\dagger(v_0 v_1 \ldots v_N)$ is zero if and only if there exists an $1 \leq i \leq N$ such that $P_i^*(v_0 v_i) = 0$, by the open-mindedness property of $P^\dagger$ (Paris, 1994, p. 95). Since (12) holds for all such assignments, these conditions fully describe the maximum entropy function $P^\dagger$.

6.3 Correctness of OBN-ccDS

We note that Step 1 of OBN-ccDS algorithm determines an OBN for $P^\dagger(v_0) = P^*(v_0)$; since the output of the algorithm agrees with Step 1 in the centre, the algorithm correctly outputs a DAG and conditional probabilities in the centre.

To establish correctness of OBN-ccDS, we have to take care of the variables in the appendices. The arguments for the two dataset case apply here in exactly the same way. We hence obtain the following:

Proposition 23. *In the case of consistent centred datasets, OBN-ccDS finds an OBN without underdetermined variables.*

In sum, then, in the centred dataset case, an OBN can be obtained directly from the datasets. There is no need to maximise entropy via (1).

7. Determining Conditional Probabilities Algebraically

As we demonstrated in the previous sections, there are natural classes of cases in which we can avoid Step 5b of OBN-cDS altogether and can parameterise an OBN just by means of Step 5a.

We saw in Figure 3 that there are cases in which a variable and its parents are not jointly measured by a single dataset. Hence, Step 5a will not yield the conditional probabilities at such a variable in such a case. Determining these probabilities by means of solving an optimisation problem appears to be inevitable.

However, it turns out that in many of these cases we can determine all conditional probabilities in an OBN without solving any optimisation problem. To compute the conditional probabilities we instead need to find the roots of a polynomial of low degree in one variable.

First, we shall investigate relatively simple cases. In Section 8, we shall use these simple cases as stepping stones to more complex situations. The computational speed-up obtained by solving an algebraic problem rather than an optimisation problem compares favourably to OBN-cDS presented in Section 3, which is already a significant improvement on the brute-force approach.

7.1 Sufficient Conditions for an Algebraic Solution

Theorem 24. *If $\mathcal{H}$ consists of $N \geq 3$ binary variables $A_0, A_1, \ldots, A_{N-1}$ such that A_0 is a child of every other A_i and such that every combination of $N-1$ variables is measured in some dataset (i.e., for all $0 \leq j \leq N-1$ there exists a dataset DS_j which measures $\{A_0, \ldots, A_{N-1}\} \setminus \{A_j\}$), then the conditional probabilities of $P^\dagger$ can be found by applications of Step 5a and by computing all roots of a polynomial $\mathcal{P}(x)$ of degree $2^{N-1}-1$ which are in $[0,1]$. This polynomial $\mathcal{P}(x)$ can be found efficiently.*

The simple case $N = 3$ will be discussed in more detail in Section 7.2. Note that this theorem holds for an arbitrary DAG structure $\mathcal{H}$ on the variables $A_1, \ldots, A_{N-1}$.

The proof, which is rather long, can be found in Appendix C.

Computational Complexity. Finding all real roots of polynomials with real coefficients can be done quickly. Already in 1970, an algorithm was reported which found all roots of polynomials ranging in degree from 20 to 50, on average in $2n^2$ milliseconds (Jenkins and Traub, 1970). Computing the polynomial $\mathcal{P}(x)$ is clearly a polynomial time problem in terms of the right-hand sides of the constraints in our optimisation problem (22).

7.2 Triangles

We shall now see that for $N = 3$, Theorem 24 allows us to simply read-off $P^\dagger$, if all variables are binary. As we show thereafter, the assumption of binary variables in Theorem 24 cannot be weakened.

7.2.1 BINARY VARIABLES

The simplest case in which we cannot avoid an under-determined variable arises as follows: Datasets DS_1, DS_2, DS_3 measure sets of binary variables $\{A_0, A_1\}$, $\{A_0, A_2\}$ and $\{A_1, A_2\}$, respectively,

with each A_i taking $a_i, \bar{a}_i$ as possible values. Furthermore, with respect to all marginal measured frequencies all pairs of measured variables are dependent. Hence, there are edges between all three variables in the graph $\mathcal{G}$. Clearly, this collection of datasets is not centred. Every acyclic orientation of the edges of $\mathcal{G}$ makes one vertex a child of the other two vertices. This variable is under-determined. An example for three such datasets can be found in Table 1.

Proposition 25 (Landes and Williamson (2016))**.** *Under the assumptions of Theorem 24 and the assumption that $N = 3$ an OBN can be obtained by simple algebraic means without solving an optimisation problem.*

We now give pseudo code for the proof; see Appendix D for a full proof.

Input: Consistent datasets DS_1, DS_2, DS_3 each measuring two variables, $\{A_0, A_1\}$, $\{A_0, A_2\}$ and $\{A_1, A_2\}$, respectively.
1) Express constraints on the unknown probabilities $P(v)$ of the eight assignments $v@\{A_0, A_1, A_2\}$ as a system of seven inhomogeneous linear equations.
2) Compute the entropy of a probability function $H(P)$ in terms of a single unknown.
3) Take the derivative of this function with respect to this unknown and equate it to zero.
4) Collect terms of derivative to obtain a polynomial of degree 3 in a single unknown.
5) Determine the possible solutions to this polynomial via basic textbook algebra.
6) Find the maximum entropy solution by testing all possible solutions.
7) Construct OBN from found maximum entropy solution.
Output: Unique OBN.

Ceteris paribus, the proof Theorem 24 has the equivalent of the first 3 steps. In Step 4, the resulting polynomial is of degree 2^{N-1}. The key idea in both proofs is to note that the constraints are only satisfied in an affine linear space of dimension one or zero. Hence, one can express the entropy $H(P)$ in terms of a single unknown probability.

Two Examples and one Remark. The question arises whether the problem structure is such that the discriminant of the polynomial, Δ, is always strictly greater than zero (equal to zero, strictly less than zero). If this were the case, then computing $P^\dagger$ would be even quicker (since one would have to check fewer possible roots of the polynomial $\mathcal{P}(x)$). Furthermore, if it were the case that $\Delta = 0$ always holds, then there might be some way of intuitively understand the values of $P^\dagger$ since x^* is then particularly simple. However, we have found cases in which $\Delta < 0, \Delta = 0$ and $\Delta > 0$, respectively, see Example 26 and Example 27 for $\Delta \neq 0$. For $\Delta < 0$ it seems particularly hard to intuitively grasp the values x^* can take. Cases with $\Delta = 0$ arise if $P_i^*(a_j^v a_k^v) = P_i^*(a_k^v)P_i^*(a_k^v)$ holds for all combinations of $v@V, i, j, k$ where defined and if also one of these measured frequencies equals zero.

Appendix E contains the source code to compute the following examples.

Example 26 (Negative Discriminant)**.** Three datasets measured the following:

$$DS_1: \quad P_1^*(a_0a_1) = \frac{55}{100} \qquad P_1^*(a_0\bar{a}_1) = \frac{10}{100} \qquad P_1^*(\bar{a}_0a_1) = \frac{33}{100} \qquad P_1^*(\bar{a}_0\bar{a}_1) = \frac{2}{100}$$
$$DS_2: \quad P_2^*(a_0a_2) = \frac{64}{100} \qquad P_2^*(a_0\bar{a}_2) = \frac{1}{100} \qquad P_2^*(\bar{a}_0a_2) = \frac{7}{100} \qquad P_2^*(\bar{a}_0\bar{a}_2) = \frac{28}{100}$$

$$DS_3: \quad P_3^*(a_1a_2) = \frac{61}{100} \qquad P_3^*(a_1\bar{a}_2) = \frac{27}{100} \quad P_3^*(\bar{a}_1a_2) = \frac{10}{100} \qquad P_3^*(\bar{a}_1\bar{a}_2) = \frac{2}{100}$$
$$P^*(a_0) = \frac{65}{100} \qquad P^*(a_1) = \frac{88}{100} \quad P^*(a_2) = \frac{71}{100}$$

Matlab computed the discriminant to be equal to $-1.097210366162764e-10$ and $P^\dagger(\bar{a}_0\bar{a}_1\bar{a}_2) = x = 0.01752872708409$.

Example 27 (Positive Discriminant)**.** Three datasets measured the following:

$$DS_1: \quad P_1^*(a_0a_1) = \frac{1}{13} \qquad P_1^*(a_0\bar{a}_1) = \frac{3}{13} \quad P_1^*(\bar{a}_0a_1) = \frac{5}{13} \qquad P_1^*(\bar{a}_0\bar{a}_1) = \frac{4}{13}$$
$$DS_2: \quad P_2^*(a_0a_2) = \frac{1}{13} \qquad P_2^*(a_0\bar{a}_2) = \frac{3}{13} \quad P_2^*(\bar{a}_0a_2) = \frac{2}{13} \qquad P_2^*(\bar{a}_0\bar{a}_2) = \frac{7}{13}$$
$$DS_3: \quad P_3^*(a_1a_2) = \frac{2}{13} \qquad P_3^*(a_1\bar{a}_2) = \frac{4}{13} \quad P_3^*(\bar{a}_1a_2) = \frac{1}{13} \qquad P_3^*(\bar{a}_1\bar{a}_2) = \frac{6}{13}$$
$$P^*(a_0) = \frac{4}{13} \qquad P^*(a_1) = \frac{6}{13} \quad P^*(a_2) = \frac{3}{13}$$

Matlab computed the discriminant to be equal to $= 1.525101176783421e-08$ and $P^\dagger(\bar{a}_0\bar{a}_1\bar{a}_2) = x = 0.27329803010335$.

Setting out, we made the assumption that *marginal frequency distributions are satisfiable by some joint probability function defined on the set V of all the variables measured by the datasets.* This might seem like an unecessarily strong assumption. Initially, one might be tempted to only assume that the marginal frequency distributions agree on joint domains. We now show that such a weaker assumption does not suffice for our purposes.

Remark 28. There exist three datasets with measured frequencies which agree on common variables which are inconsistent in the sense that there does *not* exist a probability function P defined on the entire domain which agrees with all marginal frequencies.

Proof: The measured frequencies are as follows:

$$DS_1: \quad P_1^*(a_0a_1) = \frac{55}{100} \qquad P_1^*(a_0\bar{a}_1) = \frac{10}{100} \quad P_1^*(\bar{a}_0a_1) = \frac{33}{100} \qquad P_1^*(\bar{a}_0\bar{a}_1) = \frac{2}{100}$$
$$DS_2: \quad P_2^*(a_0a_2) = \frac{64}{100} \qquad P_2^*(a_0\bar{a}_2) = \frac{1}{100} \quad P_2^*(\bar{a}_0a_2) = \frac{7}{100} \qquad P_2^*(\bar{a}_0\bar{a}_2) = \frac{28}{100}$$
$$DS_3: \quad P_3^*(a_1a_2) = \frac{70}{100} \qquad P_3^*(a_1\bar{a}_2) = \frac{18}{100} \quad P_3^*(\bar{a}_1a_2) = \frac{1}{100} \qquad P_3^*(\bar{a}_1\bar{a}_2) = \frac{11}{100}$$
$$P^*(a_0) = \frac{65}{100} \qquad P^*(a_1) = \frac{88}{100} \quad P^*(a_2) = \frac{71}{100}$$

From (23) we infer that $x_2 - x_8 = f - g = -0.1$ holds and since $x_2 \geq 0$ (we are only interested in probabilities) we obtain $x_8 \geq 0.1$. Similarly, we have $x_7 + x_8 = d = 0.02$ and since $x_7 \geq 0$ we obtain $x_8 \leq 0.02$. Contradiction.

Hence, the system of equations (23) is not solved by any probability function. This means that there is no probability function defined on $V = \{A_1, A_2, A_3\}$ which agrees with P_1^*, P_2^* and P_3^* on their respective domains. ■

In Section 5.3, we calculated the maximum entropy function $P^\dagger$ for any two datasets with measured frequencies which agree on common variables, i.e., which agree on the centre. In particular, $P^\dagger$ agrees with P_1^* and P_2^* on their respective domains. Given this, and given the above remark, we can now answer the following question: for n datasets whose measured frequency distributions agree on common variables, does there always exist a probability function defined on the entire domain which agrees with all dataset distributions? If $1 \leq n \leq 2$, then the answer is "yes", if $n \geq 3$, the answer is "no".

This remark also explains why we sampled from a probability distribution defined over the entire domain in our Matlab implementation (Section 3.4). Had we instead sampled from multiple distributions agreeing on joint domains, we would have had no guarantee that there exist a probability function consistent with all (in our sense consistent) observations.

7.2.2 Non-Binary Variables

In Proposition 25, we saw that one can determine the parameters of an OBN algebraically, if all variables are binary by inspecting the roots of a cubic polynomial. We now show that this does not extend to the case in which at least one variable is non-binary.

Let us again consider datasets DS_1, DS_2, DS_3 which measure sets of variables $\{A_1, A_2\}$, $\{A_0, A_2\}$ and $\{A_0, A_1\}$. However, now one of the variables is at least ternary. Let n_i denote the arity of A_i, the number of possible values A_i can take.

For $v@\{A_0, A_1, A_2\}$ we obtain $n_0n_1 + n_0n_2 + n_1n_2$ constraints of the form

$$\begin{aligned} P^\dagger(a_1^v a_2^v) &= P_1^*(a_1^v a_2^v) \\ P^\dagger(a_0^v a_2^v) &= P_2^*(a_0^v a_2^v) \\ P^\dagger(a_0^v a_1^v) &= P_3^*(a_0^v a_1^v). \end{aligned}$$

Restricting attention to constraints for DS_1 of the form $P^\dagger(a_1^v a_2^v) = P_1^*(a_1^v a_2^v)$ we note that none of these constraints is redundant.

On the other hand, for the constraints involving P_2^* we have for all $v@\{A_2\}$

$$\sum_{g@\{A_0\}} P^\dagger(a_0^g a_2^v) = P^\dagger(a_2^v) = P_2^*(a_2^v). \tag{13}$$

This constraint we have already from DS_1. So, we have found that n_2 constraints (arity of A_2 many) in (13) are redundant.

Similarly, for the constraints involving P_3^* we have for all $v@\{A_0\}$ and all $v'@\{A_1\}$ that

$$\begin{aligned} \sum_{g@\{A_0\}} P^\dagger(a_0^g a_1^v) &= P^\dagger(a_1^v) = P_3^*(a_1^v) \\ \sum_{g'@\{A_1\}} P^\dagger(a_0^{v'} a_1^g) &= P^\dagger(a_0^{v'}) = P_3^*(a_0^{v'}). \end{aligned}$$

From this third set of constraints we found $n_0 + n_1$ constraints which each can be expressed in terms of previously found constraints. However, not all these newly found constraints are independent. Adding these constraints we obtain

$$\sum_{v@\{A_0,A_1\}} P^\dagger(a_0^v a_1^v) = 1.$$

So, one constraint can be expressed in terms of the other n_1+n_2-1 constraints. Hence, we showed that at least $n_0+n_1+n_2-1$ constraints of the $n_0n_1+n_0n_2+n_1n_2$ constraints are redundant. So, the number of meaningful, i.e., non-redundant, constraints is at most

$$n_0n_1+n_0n_2+n_1n_2-(n_0+n_1+n_2-1) \ .$$

We now give a simple result in elementary number theory relating this number to the number of unknowns ($n_0n_1n_2$):

Proposition 29. *For natural numbers $n_0, n_1, n_2 \geq 2$ it holds that*

$$n_0n_1n_2-(n_0n_1+n_0n_2+n_1n_2-(n_0+n_1+n_2-1))=1, \textit{ if and only if } n_0+n_1+n_2=6$$
$$n_0n_1n_2-(n_0n_1+n_0n_2+n_1n_2-(n_0+n_1+n_2-1))\geq 2, \textit{ if and only if } n_0+n_1+n_2\geq 7 \ .$$

Proof: It suffices to note that

$$n_0n_1n_2-(n_0n_1+n_0n_2+n_1n_2-(n_0+n_1+n_2-1))=(n_0-1)(n_1n_2-n_1-n_2+1)$$

is strictly increasing in $n_0, n_1, n_2 \geq 2$ and that for $n_0+n_1+n_2=6$ this expression equals 1 and for $n_0+n_1+n_2=7$ this expression equals 2. ■

So, if not all variables are binary ($n_0+n_1+n_2 \geq 7$), there are at least two more unknowns than there are constraints. This shows that there is no hope for finding a version of Theorem 24 with (some) non-binary variables which reduces to an algebraic problem in a *single* unknown.

7.3 Unmeasured Substates

Theorem 24 allows us to quickly calculate certain conditional probabilities. As we saw above, if $N=3$, then, after some linear algebra, we can determine these conditional probabilities without needing to solve an optimisation problem. We shall now see that there is another special case of Theorem 24 in which the conditional probabilities can be obtained even quicker for arbitrary $N \geq 3$.

As we saw in the proof of Theorem 24: if $P^\dagger$ assigns some $v@V$ zero probability, then the set of of possible solutions consists only of a single element, $P^\dagger$. Exploiting this observation we obtain

Corollary 30. *Under the assumptions of Theorem 24, if one measured marginal frequency equals 0, then there exists a unique probability function $P^\dagger$, agreeing with all observations, which can be read off from (20).*

Proof: If one substate generated by the variables of one single dataset has not been measured, then $P^\dagger$ assigns this substate probability zero. But then all states which extend this substate are also assigned probability zero by $P^\dagger$. But this means that at least two x_j in (20) are equal to zero. Suppose $j \neq 2^N$. We can now compute x by $x_j=0=\delta_j x+v_j$. Substituting $x=-\delta_j v_j$ in (20) we obtain a system of linear equations of the form $x_i=v_i''$ with $v_i'' \in [0,1]$. Having assumed that there exists a probability function which is consistent with all measured frequencies we can now simply read off this function.

The case $j=2^N$ is similar. ■

Example 31 (Unmeasured Substate)**.** Consider datasets measuring sets of variables as detailed in Example 26 and Example 27. Also assume that one substate, a_0a_1 has not been measured, i.e., $P_1^*(a_0a_1) = 0$. $P^\dagger$ can be obtained by elementary logic by subsequently observing

$$
\begin{aligned}
&P^\dagger(a_0a_1a_2) = 0\\
&P^\dagger(a_0a_1\bar{a}_2) = 0\\
&P^\dagger(a_0\bar{a}_1a_2) = P^\dagger(a_0\bar{a}_1a_2) + P^\dagger(a_0a_1a_2) = P^\dagger(a_0a_2) = P_2^*(a_0a_2)\\
&P^\dagger(\bar{a}_0a_1a_2) = P^\dagger(a_0a_1a_2) + P^\dagger(\bar{a}_0a_1a_2) = P^\dagger(a_1a_2) = P_3^*(a_1a_2)\\
&P^\dagger(a_0\bar{a}_1\bar{a}_2) = P^\dagger(a_0\bar{a}_1) - P^\dagger(a_0\bar{a}_1a_2) = P_1^*(a_0\bar{a}_1) - P_2^*(a_0a_2)\\
&P^\dagger(\bar{a}_0a_1\bar{a}_2) = P^\dagger(\bar{a}_0a_1) - P^\dagger(\bar{a}_0a_1a_2) = P_1^*(\bar{a}_0a_1) - P_3^*(a_1a_2)\\
&P^\dagger(\bar{a}_0\bar{a}_1a_2) = P^\dagger(\bar{a}_0a_2) - P^\dagger(\bar{a}_0a_1a_2) = P_2^*(\bar{a}_0a_2) - P_3^*(a_1a_2)\\
&P^\dagger(\bar{a}_0\bar{a}_1\bar{a}_2) = P^\dagger(\bar{a}_0\bar{a}_2) - P^\dagger(\bar{a}_0a_1\bar{a}_2) = P_2^*(\bar{a}_0\bar{a}_2) - P_1^*(\bar{a}_0a_1) + P_3^*(a_1a_2).
\end{aligned}
$$

Applying this to the example in Table 1 we find that all young patients died and all old patients survived. This allows us to "complete" DS_1 and DS_3 and finally DS_2 as follows (Table 10):

Table 10: Missing values (in blue) are filled in according to the unique probability function consistent with the datasets from Table 1.

		Variables measured in Dataset 1	Variables measured in Dataset 2	
	Observation	Sex	Treatment Outcome	Age
Observations of Dataset 1	Patient 1	Male	survived	$60+$ years
	Patient 2	Female	died	$40-59$ years
	Patient 3	Female	survived	$60+$ years
Observations of Dataset 2	Patient 4	$P(\text{Male}) = P(\text{Female}) = 50\%$	survived	$60+$ years
	Patient 5	$P(\text{Male}) = P(\text{Female}) = 50\%$	survived	$60+$ years
	Patient 6	Female	died	$40-59$ years
Observations of Dataset 3	Patient 7	Female	survived	$60+$ years
	Patient 8	Male	survived	$60+$ years
	Patient 9	Female	died	$40-59$ years
		Variable measured in Dataset 3		Variable measured in Dataset 3

8. Optimisation Along Separate Paths

In the previous Section 7, we studied, in some detail, graphs with a single under-determined variable. We now show how we can use the techniques developed above to determine conditional probabilities in more complex graphs with multiple under-determined variables.

We shall now see that, if we can find a particular orientation of the arrows (this includes the choice of a root node), then determining an OBN can be reduced to *independently* determining the conditional under-determined variables.

Proposition 32. *If all under-determined variables of $\mathcal{H}$ are leaves, then the problem of computing an OBN can be reduced to independently computing the conditional probabilities at these leaves.*

Proof: Let A_k be an under-determined variable. Note that the only mention of conditional probabilities at A_k in (1) are in the term

$$\begin{aligned} &- \sum_{v@Anc'_k} P(a_k^v|Par_k^v)\log(P(a_k^v|Par_k^v)) \prod_{A_j\in Anc_k} P(a_j^v|Par_j^v) \\ &= - \sum_{v@Anc'_k} y_k^v \log(y_k^v) \prod_{A_j\in Anc_k} y_j^v \ . \end{aligned} \tag{14}$$

Ignoring summands which do not contain unknown y-parameters, the objective function of the optimisation problem can be written as

$$-\sum_{k\in\mathcal{U}} \sum_{v@Anc'_k} y_k^v \log(y_k^v) \prod_{A_j\in Anc_k} y_j^v \ .$$

Note that all y_j^v have been determined via Step 5a. Furthermore, note that we can write this sum as follows: no summand contains y_k and $y_{k'}$ for different $k, k' \in \mathcal{U}$.

The constraints also separate nicely: Note that no constraints contains y_k^v and $y_{k'}^w$ for different $k, k' \in \mathcal{U}$ and assignments v, w.

We can hence find an OBN by independently solving for all $i \in \mathcal{U}$ the following optimisation problems for non-negative y^v

$$\begin{aligned} \text{maximise: } & - \sum_{v@Anc'_i} y_i^v \log(y_i^v) \prod_{A_j\in Anc_i} y_j^v \\ \text{subject to: } & \sum_{g@\{A_i\}} y_i^{gv} = 1 \text{ for all } i \in \mathcal{U}, \\ & \sum_{s@Anc_i\setminus Par_{i:k}} y_i^{sv} \prod_{A_j\in Anc_i} y_j^{sv} = P_k^*(a_i^v Par_{i:k}^v) \text{ for all } i \in \mathcal{U} \text{ and} \\ & \text{for all maximal jointly measured sets of the form } \{A_i\}\cup Par_{i:k} \end{aligned}$$

■

If one of the under-determined leaves satisfies conditions considered in Section 7, then determining conditional probabilities simplifies even further.

Theorem 33. *If the following four equivalent conditions hold:*

- *every directed path in $\mathcal{H}$ contains at most one under-determined variable,*
- *no under-determined variable in $\mathcal{H}$ has an ancestor nor a descendant which is under-determined,*
- *no under-determined variable in $\mathcal{H}$ has an ancestor which is under-determined,*
- *no under-determined variable in $\mathcal{H}$ has a descendant which is under-determined,*

then the problem of computing an OBN can be reduced to independently computing the conditional probabilities at the under-determined variables.

As will be clear from the proof, this theorem is a simple instance of the fact that the more separate pathways the under-determined vertices are on, the more the optimisation problem can be split into independent optimisation problems (Section 4.1, Property 6).

Proof: It is clear that these four conditions are equivalent.

Let us note that were we to drop all descendants of under-determined variables we would obtain a graph in which all under-determined variables are leaves. That is, we would be in the situation investigated in Proposition 32.

Hence, the conditional probabilities at an arbitrary under-determined variable A_k appears in the entropy equation (1) in the summands for the descendants of A_k as well as in terms of the form in (14). Let A_g be a descendant of A_k. Then such a summand has the form

$$-\sum_{v@Anc'_g}\Big(\prod_{A_j\in Anc'_g} y_j^v\Big)\log(y_g^v) = -\sum_{v@Anc'_g}\Big(y_k^v \prod_{\substack{i\in\{1,\dots,g-1\}\setminus\{k\}\\ A_i\in Anc_g}} y_i^v\Big) y_g^v \log(y_g^v).$$

Note that only the y_k^v are unknown parameters and each y_k^v appears linearly in the above term. Adding all summands in which an y_k^v appears we still find that y_k^v appears linearly with a coefficient that can be directly determined by Step 5a.

Hence, the problem of computing an OBN is of the same order of computational complexity as computing the conditional probabilities at the under-determined variables independently. ■

9. Concluding Remarks

We have explored the question of how to efficiently determine an OBN from consistent datasets. We provided a general algorithm OBN-cDS which has the potential to substantially reduce the dimension of the optimisation problem, in comparison to the brute-force approach. For centred collections of datasets (in particular for any pair of datasets) we showed how to obtain an OBN without needing to solve an optimisation problem. We also discovered cases in which an OBN can be found by algebraic means and cases in which the optimisation problem can be split into independent optimisation problems.

A number avenues for further research stand out to us: (i) to identify further efficiency savings to OBN-cDS; (ii) to extend the methodology to include inconsistent datasets; and (iii) to apply the Matlab implementation to real-world datasets. More specifically, further research could address the optimisation problem we tackled from a computational point of view. For example, how could one calculate or estimate the number of local maxima? Can (the number of) maxima be characterised by the number of unknowns, the DAG of the OBN and/or the geometry of the feasible region? Can we compute some of these maxima relatively quickly? And (iv) to extend the methodology to continuous variables. Statistical matching for continuous variables is an area of active research (Little and Rubin, 2014). Obviously, discretisation is a viable strategy, if a suitable discretisation is available and the datasets are large and reliable to approximate the frequency distribution of the measured variables.

Two questions in the wider research context are also of interest: (v) to trace connections between the OBN approach and the project of explainable AI (Ras et al., 2022). In particular, how is a graphical approach employing Bayesian networks helpful for explanatory purposes and how does the maximum entropy approach compare with statistical matching techniques with respect to

explainability? (vi) to tackle the thorny methodological issue of integrating data obtained from observational study designs with data obtained from intervention studies (Hauser and Bühlmann, 2014).

Acknowledgements

We would like to thank Sylvain Barde for initial advice on the Matlab implementation and Christian Wallmann for providing some Matlab code.

Jürgen Landes gratefully acknowledges funding from the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 405961989 and 432308570, the European Research Council ('PhilPharm' grant 639276) and the UK Arts and Humanities Research Council (AH/I022957/1).

Jon Williamson gratefully acknowledges funding from the Leverhulme Trust (RPG-2019-059), the Deutsche Forschungsgemeinschaft (432308570), and the UK Arts and Humanities Research Council (AH/I022957/1).

Appendix A. Theoretical Background to OBN-cDS

Here we provide versions of results from Williamson (2005a, Sections 5.6 and 5.7) that can be used to demonstrate the correctness of the algorithm OBN-cDS.

The task is to maximise entropy subject to the constraints $P_{\downarrow V_i} = P_i^*$, for $i = 1, \ldots, h$, where V_i is the subset of variables measured by dataset DS_i. Each constraint ensures that the agent's probability function P matches a dataset distribution P_i^*. Each such constraint can be represented as set of equality constraints on probability function x of the form $\{f_i^{v_i}(x^{v_i}) := x^{v_i} - P_i^*(v_i) = 0 : v_i@V_i\}$, where $x^{v_i} := x(v_i)$. Let π be the set of all these equality constraints. We denote the feasible region (i.e., the set of constrained probability functions) by,

$$\mathbb{P}_\pi = \{x \in \mathbb{P} : f_i^{v_i}(x^{v_i}) = 0, v_i@V_i, i = 1, \ldots, h\}.$$

As usual, we assume throughout that the constraints are mutually consistent. $\mathbb{P}_\pi$ is a closed convex set of probability functions and the entropy function is strictly concave, so there is a unique function $P^\dagger$ on $\mathbb{P}_\pi$ with maximum entropy.

Form undirected graph $\mathcal{F}$ by taking variables in V as nodes and joining two nodes by an edge, if and only if they are variables measured by the same dataset.

Theorem 34. If Z separates X from Y in $\mathcal{F}$, then $X \perp\!\!\!\perp_{P^\dagger} Y \mid Z$ for the function $P^\dagger$ satisfying the constraints which maximises entropy.

Proof: The first step is to use standard Lagrange multiplier optimisation. If $x \in \mathbb{P}_\pi$ is a local maximum of H then there are constants ('multipliers') $\mu, \lambda_i^{v_i} \in \mathbb{R}$ such that

$$\frac{\partial H}{\partial x^v} + \mu + \sum_{i=1}^{h} \sum_{v_i@V_i} \lambda_i^{v_i} \frac{\partial f_i^{v_i}}{\partial x^v} = 0 \tag{15}$$

for each assignment $v@V$, where μ is the multiplier corresponding to the additivity constraint $\sum_{v@V} x^v = 1$.

Now $x_i^{v_i} = \sum_{v@V, v\sim v_i} x^v$, where the sum is over all v consistent with v_i, so

$$\frac{\partial f_i^{v_i}}{\partial x^v} = \frac{\partial f_i^{v_i}}{\partial x_i^{v_i}} \frac{\partial x_i^{v_i}}{\partial x^v} = \frac{\partial f_i^{v_i}}{\partial x_i^{v_i}}.1$$

where v_i is the assignment to V_i that is consistent with v. Furthermore,

$$\frac{\partial H}{\partial x^v} = -1 - \log x^v,$$

so Equation 15 can be written

$$\log x^v = -1 + \mu + \sum_{i=1}^{h} \lambda_i^{v_i} \frac{\partial f_i^{v_i}}{\partial x_i^{v_i}}$$

where v_i is the assignment to V_i that is consistent with v. Thus,

$$x^v = e^{\mu - 1} \prod_{i=1}^{h} e^{\lambda_i^{v_i} \frac{\partial f_i^{v_i}}{\partial x_i^{v_i}}} \tag{16}$$

Hence the local maximum x is representable as a product of functions, each of which depends only on variables in the V_i (the leading term is a constant). The probability function x is said to *factorise* according to the sets $V_1, \ldots V_h$, and since these sets are complete subsets of $\mathcal{F}$, x is said to *factorise* according to $\mathcal{F}$ (Lauritzen, 1996, pp. 34–35). The *global Markov condition* says that if Z separates X from Y in $\mathcal{F}$,then $X \perp\!\!\!\perp Y \mid Z$, and this condition is a straightforward consequence of factorisation according to $\mathcal{F}$ (Lauritzen, 1996, Proposition 3.8). Thus the theorem follows for all local maxima, and in particular for the global maximum $P^\dagger$. ■

Step 2 of OBN-cDS creates an undirected graph $\mathcal{G}$ that is a subgraph of $\mathcal{F}$. This replaces each complete graph on V_i by the Markov network structure $\mathcal{G}_i$ learned at Step 1. This replacement clearly keeps the above property intact, i.e., if Z separates X from Y in $\mathcal{G}$ then $X \perp\!\!\!\perp_{P^\dagger} Y \mid Z$ for any $P^\dagger$ satisfying the constraints which maximises entropy.

Steps 3 and 4 produce a directed acyclic graph $\mathcal{H}$ from $\mathcal{G}$. D-separation (see Pearl, 1988, §3.3.1) allows one to read off probabilistic independencies from $\mathcal{H}$:

Theorem 35. *If Z D-separates X from Y in the directed acyclic graph $\mathcal{H}$ then $X \perp\!\!\!\perp_{P^\dagger} Y \mid Z$ for the function $P^\dagger$ satisfying the constraints which maximises entropy.*

Proof: Since $\mathcal{G}^T$ is triangulated, the ordering yielded by maximum cardinality search is a *perfect* ordering (for each vertex, the set of its adjacent predecessors is complete in the graph) (Neapolitan, 1990, Theorem 3.2). Because the cliques are ordered according to highest labelled vertex where the vertices have a perfect ordering, the clique order has the *running intersection property* (for each clique, its intersection with the union of its predecessors is contained in one of its predecessors) (Neapolitan, 1990, Theorem 3.1). Now $P^\dagger$ factorises according to the cliques of $\mathcal{G}^T$, since it factorises according to $V_1, \ldots, V_h$ and these sets are complete in $\mathcal{G}^T$ and so are subsets of its cliques. These three facts imply that $P(v) = \prod_{i=1}^{l} P(f_i^v | e_i^v)$ for each $v@V$, where f_i^v, e_i^v are the assignments to F_i, E_i respectively which are consistent with v (Neapolitan, 1990, Theorem 7.4).

Take an arbitrary component $P(f_i^v|e_i^v)$ of this factorisation. Each member of E_i is a parent (in $\mathcal{H}$) of each member of F_i and the members of F_i form a complete subgraph of $\mathcal{H}$ so we can write $F_i = \{A_{i_1}, \ldots, A_{i_k}\}$ where the parents of A_{i_j} are $Par_{i_j} \stackrel{\text{df}}{=} E_i \cup \{A_{i_1}, \ldots, A_{i_{j-1}}\}$. Hence,

$$\begin{aligned} P(f_i^v|e_i^v) &= P(a_{i_1} \cdots a_{i_k}|e_i) \\ &= \prod_{j=1}^{k} P(a_{i_j}|e_i a_{i_1} \cdots a_{i_{j-1}}) \\ &= \prod_{j=1}^{k} P(a_{i_j}|par_{i_j}), \end{aligned}$$

where a_{i_j} and par_{i_j} are the assignments to A_{i_j}, Par_{i_j} respectively that are consistent with v. Furthermore, each variable V_i occurs in precisely one F_j, so

$$P(v) = \prod_{i=1}^{n} P(a_i|par_i) \tag{17}$$

for each $v@V$. When (17) holds, P is said to *factorise* with respect to $\mathcal{H}$, and $\mathcal{H}$ together with the specified values of $P(a_i|par_i)$ form a Bayesian network. It follows that if Z D-separates X from Y in $\mathcal{H}$, then $X \perp\!\!\!\perp_{P^\dagger} Y \mid Z$ (see Neapolitan, 1990, Theorem 6.2). ■

Thus the algorithm does indeed correctly produce a graph $\mathcal{H}$ that can feature as the graph of a Bayesian network representation of the maximum entropy function $P^\dagger$. See §3.3 for justification of Step 5 of the algorithm, which obtains the parameters of the Bayesian network.

Appendix B. Proof of Propositions 15, 16 and 17

Proposition 15. *$\mathcal{G}^+$ is healthy and $\mathcal{G}^+$ is a triangulation of $\mathcal{G}$ which, on input $\mathcal{G}$, can be computed in polynomial time.*

Proof: In no step have we added an unhealthy edge. Hence, $\mathcal{G}^+$ is healthy.

First, note that all fill-in edges added to $\mathcal{G}$ are chords of simple cycles of $\mathcal{G}$ of length four or greater.

Furthermore, note that no fill-in edge computed in Step III connects two vertices in the same set $N(A^i)$, since we added all these chords in Step II. Likewise, no fill-in edge computed in Step IV connects two vertices in the same set $N(A^i)$, since we added all these chords in Step II.

We next show that $\mathcal{G}^+$ is triangulated. Let c be a simple cycle in $\mathcal{G}^+$ which contains at least four vertices. We need to show that it possesses a chord.

If c does not contain a vertex in an appendix, then no edge of this cycle has been computed in Step III. So, all its edges are in $\mathcal{G}^-$. Hence, a chord was computed in Step IV and added in Step V.

If c lies fully within $A^i \cup N(A^i)$ for some i, then no edge nor chord of c has been computed in Step IV and added in Step V. So, all its edges are in $\mathcal{G}^-$. Hence, a chord was computed in Step I or Step III and added in Step II or Step V.

Finally, suppose that c contains a vertex A in some appendix A^i and a vertex B not in $A^i \cup N(A^i)$. Recall that $N(A^i)$ separates A^i from $V \setminus (A^i \cup N(A^i))$. Since c is a simple cycle and

$N(A^i)$ is a separator, c has to contain two different vertices $B_1, B_2 \in N(A^i)$. An edge between B_1, B_2 is either in $\mathcal{G}$ or it has been computed in Step I and added in Step II.

To complete the proof it suffices to observe that $\mathcal{G}$ and $\mathcal{G}^+$ are undirected graphs on V and that the latter contains every edge contained in the former. Hence, $\mathcal{G}^+$ is a triangulation of $\mathcal{G}$.

Given $\mathcal{G}$, $\mathcal{G}^+$ can be computed in polynomial time, since computing a minimal triangulation is achievable in polynomial time (Heggernes, 2006). ■

Proposition 16. *$\mathcal{G}^{++}$ is a healthy and minimal triangulation of $\mathcal{G}$ which can be computed in polynomial time.*

Proof: $\mathcal{G}^{++}$ consists of edges contained in $\mathcal{G}^+$ and of edges within the $N(A^i)$. Hence, $\mathcal{G}^{++}$ is healthy.

From the above proposition we know that $\mathcal{G}^+$ is a triangulation of $\mathcal{G}$. Recall that all edges added in Step II connect two variables in $N(A^i)$. We can hence successively add such edges to $\mathcal{G}_0$ until we obtain a triangulation of $\mathcal{G}_0$. Hence, $\mathcal{G}^{++}$ is triangulated. Since $\mathcal{G}^{++}$ contains all the edges of $\mathcal{G}$ it follows that $\mathcal{G}^{++}$ is a triangulation of $\mathcal{G}$.

We now turn to the minimality claim. All edges in $\mathcal{G}^+$ which are not in $\mathcal{G}$ are chords of simple cycles of $\mathcal{G}$. In Step III and Step IV we computed further chords which were minimal in the following sense: After adding chords to $\mathcal{G}$ in Step II, dropping any subset of chords computed in Step III and Step IV would have resulted in a non-triangulated graph (recall that the triangulations in Step III and Step IV were *minimal*). Hence, $\mathcal{G}_0$ is either a minimal triangulation of $\mathcal{G}$—in which case our algorithm terminates—, or $\mathcal{G}_0$ is a strict sub-graph of a minimal triangulation of $\mathcal{G}$.

For the latter case recall that $\mathcal{G}^+$ is a healthy triangulation of $\mathcal{G}$ which strictly contains $\mathcal{G}_0$. Furthermore, every edge in $\mathcal{G}^+$ which is not in $\mathcal{G}_0$ connects two variables in $N(A^i)$. Hence, adding such edges successively, we eventually reach a triangulated graph.

VII: We can run, for instance, Berry's algorithm; see Berry (1999); but only add fill-in edges with start and end point in some $N(A^i)$. We then stop since we have found the minimal triangulation $\mathcal{G}^{++}$.

Clearly, $\mathcal{G}^{++}$ can be computed in polynomial time. ■

Proposition 17. *We can find an OBN in which all arrows beginning in an appendix also end in this same appendix. Furthermore, every variable in an appendix is fully determined.*

Proof: We begin by executing Steps 1–3 of OBN-cDS. Next, further undirected edges to $\mathcal{G}$ are added: for all datasets DS_i connect all variables measured in DS_i which are not in the appendix A^i by an edge, if they not already connected by an edge. Note that the resulting undirected graph $\mathcal{G}'$ is healthy.

Next, we compute a minimal healthy triangulation $\mathcal{G}^T$ of $\mathcal{G}'$ (Step 3). The existence of such a triangulation is shown in Proposition 16. For the remainder of this proof we shall focus on an arbitrary connected component of $\mathcal{G}^T$.

In Step 5, we chose an enumeration which first enumerates all variables measured in DS_i which are not in A^i and only afterwards enumerates the variables in A^i. This is possible for the following reasons: The variable $A \in DS_i$ appearing first in the enumeration is not in the appendix A^i, unless we begin our enumeration in A^i. However, we can clearly choose to begin the enumeration elsewhere in the connected component of $\mathcal{G}^T$. Recall that the variables measured in DS_i which are not

in A^i are all connected by edges (they form a complete sub-graph). For every fixed set $S \subset V$ of variables measured in DS_i containing A such that $S \cap A^i = \emptyset$ and every variable $B \in DS_i \setminus \{S, A^i\}$ it holds that B is connected to all variables in S. For every variable $A_i \in A^i$ it holds that B has more or equally many neighbours in S than B. We can hence chose to execute a maximum cardinality search which enumerates B prior to all variables in the appendix A^i.

It follows that all arrows connecting A^i with a variable outside of A^i point towards A^i. These arrows all originate from some variable measured in DS_i. We can hence determine the conditional probabilities at the variables in appendix A^i directly from DS_i as explained in Section 3.3.3.

Determine all other conditional probabilities as described in Step 5.

This procedure correctly determines an OBN: it differs from OBN-cDS by adding further edges and the choice of a particular enumeration. Neither modification changes the fact that the output Bayesian network is an OBN. ■

Appendix C. Proof of Theorem 24

Theorem 24. *If $\mathcal{H}$ consists of $N \geq 3$ binary variables $A_0, A_1, \ldots, A_{N-1}$ such that A_0 is a child of every other A_i and such that every combination of $N-1$ variables is measured in some dataset (i.e., for all $0 \leq j \leq N-1$ there exists a dataset DS_j which measures $\{A_0, \ldots, A_{N-1}\} \setminus \{A_j\}$), then the conditional probabilities of $P^\dagger$ can be found by applications of Step 5a and by computing all roots of a polynomial $\mathcal{P}(x)$ of degree $2^{N-1}-1$ which are in $[0,1]$. This polynomial $\mathcal{P}(x)$ can be found efficiently.*

The simple case $N = 3$ is discussed in more detail in Section 7.2.

Note that this theorem holds for an arbitrary DAG structure $\mathcal{H}$ on the variables $A_1, \ldots, A_{N-1}$.

Proof: The proof consists of two parts. In the first part, we show that the problem of computing the conditional probabilities reduces to maximising a function of a single variable. In the second part, we show that this function is a polynomial of degree less or equal to $2^{N-1}-1$.

First of all, note that A_0 is the only under-determined variable since DS_0 measures all parents of A_0. Hence, the conditional probabilities at $A_1, \ldots, A_{N-1}$ can directly be obtained by applications of Step 5a since there exists a dataset which jointly measures all these variables. This leaves us with the task of determining the conditional probabilities at the under-determined variable A_0.

There are 2^N states generated by the binary variables $A_0, \ldots A_{N-1}$. So, in order to compute the conditional probabilities at A_0 we determine probabilities of $2^N - 1$ states. The probability of the last remaining state follows from the fact that the sum over the probabilities of all states has to equal one.

We denote these states as follows:

$$\omega_1 := a_0 \wedge \bigwedge_{i=1}^{N-1} a_i$$

$$\omega_2 := a_0 \wedge \bar{a}_{N-1} \wedge \bigwedge_{i=1}^{N-2} a_i$$

$$\omega_3 := a_0 \wedge \bar{a}_{N-2} \wedge a_{N-1} \wedge \bigwedge_{i=1}^{N-3} a_i$$

$$\omega_4 := a_0 \wedge \bar{a}_{N-2} \wedge \bar{a}_{N-1} \wedge \bigwedge_{i=1}^{N-3} a_i$$

and so on.

To simplify notation we define the unknowns, probabilities of states, as $x_u := P^\dagger(\omega_u)$ for all $1 \leq u \leq 2^N$. Let $a_i^0 := \bar{a}_i$ and $a_i^1 := a_i$.

Furthermore, define the states generated by variables measured in DS_j for all $0 \leq j \leq N-1$

$$\Psi_j := \{a_0^{\epsilon_0} \wedge a_1^{\epsilon_1} \wedge \ldots \wedge a_{j-1}^{\epsilon_{j-1}} \wedge a_{j+1}^{\epsilon_{j+1}} \wedge \ldots \wedge a_{N-1}^{\epsilon_{N-1}} \ : \epsilon_i \in \{0,1\}, i \in \{0,\ldots,N-1\} \setminus \{j\}\}.$$

To simplify notation we enumerate for all $1 \leq q \leq 2^{N-1}$ the $\psi^j \in \Psi_j$ as follows: ψ_1^j is the $\psi \in \Psi_j$ for which all ϵ are equal to one, i.e., ψ_1^j does not contain a negation symbol. In ψ_2^j all ϵ but the last equal 1, i.e., only the last variable is negated. In $\psi_{2^{N-1}}^j$ all ϵ_i are equal to 0.

For all $0 \leq j \leq N-1$ and all $\psi_q^j \in \Psi_j$ $(1 \leq q \leq 2^{N-1})$ it needs to hold that

$$P^\dagger(a_j \wedge \psi_q^j) + P^\dagger(\bar{a}_j \wedge \psi_q^j) = P^\dagger(\psi_q^j) = P_j^*(\psi_q^j) \tag{18}$$

since the marginal frequency of $\psi_q^j \in \Psi_j$ has been measured in DS_j.

Observe that for all probability functions P which satisfy the constraints in (18) for all j and all $\psi^j \in \Psi_j$ are calibrated to the evidence, i.e., such a P satisfies all constraints. In other words, there are no further constraints $P^\dagger$ needs to satisfy.

From DS_{N-1} measuring $\{A_0,\ldots,A_{N-2}\}$ we obtain for all $0 \leq s \leq 2^{N-1}-1$ the following constraints on $P^\dagger$

$$\begin{aligned} x_{2s+1} + x_{2s+2} =& P^\dagger(\omega_{2s+1}) + P^\dagger(\omega_{2s+2}) \\ =& P^\dagger(\psi_{s+1}^{N-1} \wedge a_{N-1}) + P^\dagger(\psi_{s+1}^{N-1} \wedge \bar{a}_{N-1}) \\ =& P^\dagger(\psi_{s+1}^{N-1}) = P_{N-1}^*(\psi_{s+1}^{N-1}) \ . \end{aligned}$$

From DS_{N-2} measuring $\{A_0,\ldots,A_{N-3},A_{N-1}\}$ we obtain for all $0 \leq s \leq 2^{N-2}-1$ the following constraints on $P^\dagger$

$$\begin{aligned} x_{2^2s+1} + x_{2^2s+3} =& P^\dagger(\omega_{2^2s+1}) + P^\dagger(\omega_{2^2s+3}) \\ =& P^\dagger(\psi_{2s+1}^{N-2} \wedge a_{N-2}) + P^\dagger(\psi_{2s+1}^{N-2} \wedge \bar{a}_{N-2}) \\ =& P^\dagger(\psi_{2s+1}^{N-2}) = P_{N-2}^*(\psi_{2s+1}^{N-2}) \end{aligned}$$

and

$$\begin{aligned} x_{2^2s+2} + x_{2^2s+4} =& P^\dagger(\omega_{2^2s+2}) + P^\dagger(\omega_{2^2s+4}) \\ =& P^\dagger(\psi_{2s+2}^{N-2} \wedge a_{N-2}) + P^\dagger(\psi_{2s+2}^{N-2} \wedge \bar{a}_{N-2}) \\ =& P^\dagger(\psi_{2s+2}^{N-2}) = P_{N-2}^*(\psi_{2s+2}^{N-2}) \ . \end{aligned}$$

For the constraints arising from DS_0, DS_1 we can write the matrix consisting of the left hand sides of these two sets of constraints as

$$M = \begin{array}{c|c|c|c|c} A & O & O & \ldots & O \\ \hline O & A & O & \ldots & O \\ \hline O & O & A & \ldots & O \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline O & O & O & \ldots & A \end{array}$$

where

$$A = \begin{array}{|c|c|c|c|}\hline 1 & 1 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 \\ \hline 1 & 0 & 1 & 0 \\ \hline 0 & 1 & 0 & 1 \\ \hline \end{array}$$

of which there are $\frac{N}{4}$ matrices. O is the four-dimensional zero-matrix:

$$O = \begin{array}{|c|c|c|c|}\hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array} \ .$$

Note that we can transform A into the equivalent A' as follows (the last row equals the sum of the first two rums minus the third row)

$$A' = \begin{array}{|c|c|c|r|}\hline 1 & 0 & 0 & -1 \\ \hline 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 1 & 1 \\ \hline 0 & 0 & 0 & 0 \\ \hline \end{array} \ .$$

For the remainder we drop the last row which represents the vacuous constraint $0 = 0$.

We hence obtain a set of left-hand sides of the constraints, which we write as a matrix M', which is equivalent to M

$$M' = \begin{array}{|c|c|c|c|c|}\hline A' & O & O & \dots & O \\ \hline O & A' & O & \dots & O \\ \hline O & O & A' & \dots & O \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline O & O & O & \dots & A' \\ \hline \end{array} \ .$$

We now show by an inductive argument on the number of variables N that the constraints of equalling measured marginal frequencies gives rise to a system of $2^N - 1$ linear inhomogeneous equations. In matrix notation, the left-hand-side of these equations can be represented by

$$M_{\text{final}} = \begin{array}{|c|c|c|c|c|c|c|}\hline 1 & 0 & 0 & 0 & \dots & 0 & \pm 1 \\ \hline 0 & 1 & 0 & 0 & \dots & 0 & \pm 1 \\ \hline 0 & 0 & 1 & 0 & \dots & 0 & \pm 1 \\ \hline 0 & 0 & 0 & 1 & \dots & 0 & \pm 1 \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline 0 & 0 & 0 & 0 & \dots & 1 & 1 \\ \hline \end{array} \ .$$

The column vector consisting of the first $\frac{2^N}{2} - 1$ entries in the last row is the -1-multiple of the column vector formed out of the last $\frac{2^N}{2} - 1$ entries in the last column. The entry in between equals $+1$.

Our argument proceeds in $N-2$ further steps. At each further stage j, constraints arising from datasets $DS_{N-1}, \ldots, DS_{N-j-2}$ are taken into account. Constraints arising from DS_{N-j-2} are added to the system of left-hand-sides. The equation system at stage j takes the form

$$M_j = \begin{array}{|c|c|c|c|c|} \hline A_j & O & O & \ldots & O \\ \hline O & A_j & O & \ldots & O \\ \hline O & O & A_j & \ldots & O \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline O & O & O & \ldots & A_j \\ \hline \end{array} \ ,$$

where A_j is a matrix with $2^{j+1}-1$ rows and 2^{j+1} columns. The first $2^{j+1}-1$ and $2^{j+1}-1$ columns are the identity matrix. The condition on the last column applies to A_j with $j+1$ replacing N.

First step: $j=1$**,** $DS_{N-2-j} = DS_{N-3}$**.**
From DS_{N-3} measuring $\{A_0, A_1, \ldots, A_{N-4}, A_{N-2}, A_{N-1}\}$ we obtain for all $0 \leq s \leq 2^{N-3}-1$ the following constraints on $P^\dagger$ (there are further constraints arising which are considered separately below)

$$\begin{aligned} x_{2^3 s+2^2} + x_{2^3 s+2^3} =& P^\dagger(\omega_{2^3 s+2^2}) + P^\dagger(\omega_{2^3 s+2^3}) \\ =& P^\dagger(\psi^{N-3}_{2^2 s+2^2} \wedge a_{N-3}) + P^\dagger(\psi^{N-3}_{2^2 s+2^2} \wedge \bar{a}_{N-3}) \\ =& P^\dagger(\psi^{N-3}_{2^2 s+2^2}) = P^*_{N-3}(\psi^{N-3}_{2^2 s+2^2}) \ . \end{aligned}$$

We now group matrices of type A' in M' pairs, the first pair are the two uppermost A', the next pair are the third and fourth uppermost A', and so on. The constraints arising from DS_{N-3} can be represented by the fourth row in the following matrix

$$A'_1 = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ \hline \end{array} .$$

After some trivial matrix algebra we find an equivalent matrix A_1

$$A_1 = \begin{array}{|c|c|c|c|c|c|c|c|} \hline 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ \hline 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \\ \hline 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ \hline 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 & 0 & -1 \\ \hline 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ \hline \end{array}$$

and so

$$M_1 = \begin{array}{c|c|c|c|c} A_1 & O & O & \dots & O \\ \hline O & A_1 & O & \dots & O \\ \hline O & O & A_1 & \dots & O \\ \hline \vdots & \vdots & \vdots & \ddots & \vdots \\ \hline O & O & O & \dots & A_1 \end{array} .$$

This concludes the proof of the first step, $j = 2$.

General step: $j \geq 2$**,** DS_{N-2-j}**.**
Note that for all $0 \leq s \leq 2^{N-j} - 1$ we obtain the following constraints from DS_{N-2-j} measuring $\{A_0, A_1, \dots, A_{N-1}\} \setminus \{A_{N-2-j}\}$, on $P^\dagger$

$$\begin{aligned} x_{2^{j+1}s+2^j} + x_{2^{j+1}s+2^{j+1}} =& P^\dagger(\omega_{2^{j+1}s+2^j}) + P^\dagger(\omega_{2^{j+1}s+2^{j+1}}) \\ =& P^\dagger(\psi^{N-2-j}_{2^j s+2^j} \wedge a_j) + P^\dagger(\psi^{N-2-j}_{2^j s+2^j} \wedge \bar{a}_j) \\ =& P^\dagger(\psi^{N-2-j}_{2^{j+1}s+2^j}) = P^*_{N-2-j}(\psi^{N-2-j}_{2^{j+1}s+2^j}) \ . \end{aligned}$$

Adding such a constraint to the two matrices of type A_{j-1} in M_{j-1} which concern the unknowns $x_{2^{j+1}s+1}, \dots, x_{2^{j+1}s+2^{j+1}}$ we obtain the following left-hand sides:

$$\begin{array}{c|c|c|c|c|c|c|c|c|c|c|c|c} 1 & 0 & 0 & 0 & \dots & 0 & \pm 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \hline 0 & 1 & 0 & 0 & \dots & 0 & \pm 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & \dots & 0 & \pm 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \hline 0 & 0 & 0 & 1 & \dots & 0 & \pm 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & 0 & 0 & 0 & 0 & \ddots & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & \dots & 1 & 1 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \hline \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{1} \\ \hline 0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 & \pm 1 \\ \hline 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 & 0 & 0 & \dots & 0 & \pm 1 \\ \hline 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 1 & 0 & \dots & 0 & \pm 1 \\ \hline 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 1 & \dots & 0 & \pm 1 \\ \hline \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \hline 0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & 1 \end{array} .$$

The constraint arising from DS_j is set in bold.

Now it is straightforward to transform these left-hand sides of a system of linear inhomogeneous equations into the desired form

$$\begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}
\hline
1 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \mp 1 \\
\hline
0 & 1 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \mp 1 \\
\hline
0 & 0 & 1 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \mp 1 \\
\hline
0 & 0 & 0 & 1 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & \mp 1 \\
\hline
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \mp 1 \\
\hline
0 & 0 & 0 & 0 & \dots & 1 & 0 & 0 & 0 & 0 & 0 & \dots & 0 & -1 \\
\hline
\mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \dots & \mathbf{0} & \mathbf{1} \\
\hline
0 & 0 & 0 & 0 & \dots & 0 & 0 & 1 & 0 & 0 & 0 & \dots & 0 & \pm 1 \\
\hline
0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 1 & 0 & 0 & \dots & 0 & \pm 1 \\
\hline
0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 1 & 0 & \dots & 0 & \pm 1 \\
\hline
0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 1 & \dots & 0 & \pm 1 \\
\hline
\vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
\hline
0 & 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & 0 & \dots & 1 & 1 \\
\hline
\end{array} . \tag{19}$$

The condition on the last column clearly holds.

We hence obtain the matrix representation M_j as required.

Continuing this process until finally $j = N - 2$ we obtain M_{final}.

A constraint concerning a $\psi^{N-j-2} \in \Psi_{N-j-2}$ for $N - 2 \geq j \geq 0$, arising from calibrating to measured frequencies measured in DS_{N-j-2}, that we have not yet considered has a left-hand side of the form: $x_r + x_{r+2^j}$ such that there exists some natural number m with $m2^{l+1} + 1 \leq r < r + 2^j \leq (m + 1)2^{j+1}$. Summing rows r and $r + 2^j$ of M_{final} we obtain $x_r + x_{r+2^j}$. Note there is no x_{2^N}-term since x_r appears in the top half of some A_{N-j-2} and x_{r+2^j} in the bottom half of this A_{N-j-2} (2^j rows further down). When we add these rows, the x_{2^N}-terms have different signs and hence cancel out. That is, the left hand side of the not yet considered constraint is present in our system of equations represented by M_{final}.

Since all datasets are consistent, the right-hand sides of the constrained arising from ψ^l and the sum of these rows have to agree. We have hence expressed all constraints as a system of linear inhomogeneous equations in 2^N variables. This means that M_{final} represents all meaningful constraints on $P^\dagger$—other than $P^\dagger \in \mathbb{P}$.

We can hence use a single parameter, x say, to parametrise the entire solution set. This concludes the first part of the proof.

Second Part. We now identify the set of probability functions on $\omega_1, \dots, \omega_{2^N}$ with the subset of points in $\mathbb{R}^{2^N}$ whose components are non-negative and sum to 1, this simplex is denoted by $\mathbb{P}$. We can now infer from M_{final} that the set of points in $\mathbb{R}^{2^N}$ which solve all equations simultaneously is a hyperplane of dimension one, a line, $\mathcal{L}$. Since all datasets are consistent, there has to exist at least one probability function in $\mathcal{L}$. The set of probability functions consistent with the evidence is the intersection of $\mathcal{L}$ and this simplex.

For convenience, we let $x := x_{2^N} = P^\dagger(\bigwedge_{i=0}^{N-1} \bar{a}_i)$. We now express all other unknowns in terms of x by using M_{final}. Let $\vec{v}$ be the column vector consisting of the right-hand sides of the system of equations with left hand side M_{final}. For all $1 \leq j \leq 2^{N-1} - 1$ we obtain

$$x_j = \delta_j x + v_j$$

$$\begin{aligned} x_{j+2^{N-1}} &= -\delta_j x + v_{j+2^{N-1}} \\ x_{2^{N-1}} &= -x + v_{2^{N-1}} \\ x_{2^N} &= x \ , \end{aligned} \tag{20}$$

where $\delta_j \in \{-1, +1\}$. For later use we now let $D \subset \{1, \ldots, 2^{N-1} - 1\}$ be the index set with $\delta_j = +1$ and $E \subset \{1, \ldots, 2^{N-1} - 1\}$ be the index set with $\delta_j = -1$. Note that $|D \cup E| = 2^{N-1} - 1$.

If there exist two different points $\vec{l}, \vec{l'} \in \mathcal{L}$ that have the same x_{2^N} coordinate, $l_{2^N} = l'_{2^N}$, then it has to hold for all $1 \leq j \leq 2^N - 1$ that

$$l_j = v_j \pm \delta_j l_{2^N} = v_j \pm \delta_j l'_{2^N} = l'_j \ .$$

And so $\vec{l} = \vec{l'}$. Contradiction.

On the other hand, suppose there exist different $\vec{l}, \vec{l'} \in \mathcal{L}$ which have the same x_j coordinate for some $1 \leq j \leq 2^{N-1} - 1$, $l_j = l'_j$. But this means that $\vec{l}$ and $\vec{l'}$ have the same last coordinate $l_{2^N} = l'_{2^N}$. We already saw that this cannot be so. The case for $2^{N-1} \leq j \leq 2^N - 1$ is similar.

For later use we note that, this means that no coordinate is constant along the line $\mathcal{L}$, if $\mathbb{P} \cap \mathcal{L}$ contains two or more points. We can hence use any coordinate in $\mathbb{R}^{2^N}$ as a parameter to parametrise $\mathcal{L}$. We shall use $x = x_{2^N}$ as our coordinate of $\mathcal{L}$.

Geometry of the solution set of M_{final}: There are two cases of how $\mathbb{P}$ and $\mathcal{L}$ can be situated.

First case: $\mathbb{P} \cap \mathcal{L}$ does not contain an interior point of $\mathbb{P}$. That is, $\mathbb{P} \cap \mathcal{L}$ is a subset of the boundary of $\mathbb{P}$.

If $\mathbb{P} \cap \mathcal{L}$ contained two different points $\vec{l}, \vec{l'}$, then for all $\lambda \in (0, 1)$ it has to hold that $\lambda\vec{l} + (1 - \lambda)\vec{l'} \in \mathbb{P} \cap \mathcal{L}$. All these convex combinations have to be points on the boundary of $\mathbb{P}$. Since they lie on a line segment in the boundary of $\mathbb{P}$ there has to exists some index $1 \leq j \leq 2^N$ such that $(\lambda\vec{l} + (1 - \lambda)\vec{l'})_j = 0$. Hence, $l_j = l'_j$. Recall, that we showed above that two different elements of $\mathbb{P}$ have to differ on *all* coordinates. We have hence derived a contradiction.

If $\mathbb{P} \cap \mathcal{L}$ consists of a single point, then this point is, of course, $P^\dagger$. Computing whether $\mathbb{P} \cap \mathcal{L}$ consists of a single point is the case can be done very quickly. For all $1 \leq j \leq 2^N$ we check whether there exists a probability function which solves the constraints with $x_j = 0$. This can be done by solving (20). We hence do *not* have to solve an optimisation problem, all we need to do is check the very simple system of equations (20) at most 2^N times.

So, the first case only consists of one sub-case which is trivial. The second case is not so simple.

Second case: $\mathbb{P} \cap \mathcal{L}$ contains an interior point of $\mathbb{P}$. This means that there exists some probability function consistent with the constraints which assigns all $\omega_1, \ldots, \omega_{2^N}$ non-zero probability. The maximum entropy function is well-known to assign non-zero probabilities whenever possible, see for example Paris and Vencovská (1997). Hence, the maximum entropy function of $\mathcal{L}$ is non-zero everywhere on $\omega_1, \ldots, w_{2^N}$.

The entropy of a probability function on the solution set of M_{final} can now be expressed in terms of the single unknown x as

$$\begin{aligned} H(x) = &\sum_{d \in D} -(x + v_d)\log(x + v_d) - (-x + v_{d+2^{N-1}})\log(-x + v_{d+2^{N-1}}) \\ &+ \sum_{e \in E} -(-x + v_e)\log(-x + v_e) - (x + v_{e+2^{N-1}})\log(x + v_{e+2^{N-1}}) \end{aligned}$$

$$- (-x + v_{2^{N-1}}) \log(-x + v_{2^{N-1}}) - x \log(x) \ . \tag{21}$$

In the current, second, case, we know that the maximum entropy function does not assign zero probabilities. Hence, where the entropy is maximal in $\mathbb{P} \cap \mathcal{L}$ the above terms between the above brackets are all in the open interval $(0, 1)$. So, all logarithms are well-defined and the first derivative of (21) with respect to $x \in (0, 1)$ exists and is continuous in a neighbourhood of the absolute maximum.

Taking the derivative with respect to x we obtain (recall our convention of a logarithm with base e in Definition 2 to simplify the derivative of $H(x)$ with respect to x)

$$\begin{aligned}
\frac{d}{dx} H(x) =& \sum_{d \in D} -(1 + \log(x + v_d)) + 1 + \log(-x + v_{d+2^{N-1}}) \\
&+ \sum_{e \in E} 1 + \log(-x + v_e) - (+1 + \log(x + v_{e+2^{N-1}})) \\
&+ 1 + \log(-x + v_{2^{N-1}}) - (1 + \log(x)) \\
=& \sum_{d \in D} \log(-x + v_{d+2^{N-1}}) + \sum_{e \in E} \log(-x + v_e) \\
&+ \sum_{d \in D} - \log(x + v_d) + \sum_{e \in E} - \log(x + v_{e+2^{N-1}}) \\
&+ \log(-x + v_{2^{N-1}}) - \log(x) \\
=& \log((-x + v_{2^{N-1}}) \cdot \prod_{d \in D} (-x + v_{d+2^{N-1}}) \cdot \prod_{e \in E} (-x + v_e)) \\
&- \log((x) \cdot \prod_{d \in D} (x + v_d) \cdot \prod_{e \in E} (x + v_{e+2^{N-1}})) \\
=& \log\Big(\frac{-x + v_{2^{N-1}}}{x} \cdot \prod_{d \in D} \frac{-x + v_{d+2^{N-1}}}{x + v_d} \cdot \prod_{e \in E} \frac{-x + v_e}{x + v_{e+2^{N-1}}} \Big) \ .
\end{aligned}$$

This expression equals 0, if and only if the term in between the brackets equals 1. This is the case, if and only if the nominator equals the denominator. This happens, if and only if

$$\begin{aligned}
0 =& (-x + v_{2^{N-1}}) \cdot \prod_{d \in D} (-x + v_{d+2^{N-1}}) \cdot \prod_{e \in E} (-x + v_e) \\
&- (x) \cdot \prod_{d \in D} (x + v_d) \cdot \prod_{e \in E} (x + v_{e+2^{N-1}}) =: \mathcal{P}(x) \ .
\end{aligned} \tag{22}$$

Both products comprise of 2^{N-1} factors (recall that $|D \cup E| = 2^{N-1} - 1$) which is an even number. Expressing each product as a sum we obtain two polynomials of degree 2^{N-1}. Both polynomials in x have a leading coefficient of $+1$ ($1 \cdot x^{2^{N-1}}$). Hence, the difference of these polynomials is a polynomial $\mathcal{P}(x)$ of maximal degree $2^{N-1} - 1$. The exact degree of the polynomial may be even lower, if events conspire so that further terms cancel out. Whether this happens or not depends on the numerical values of the coefficients v_d, which represent the right hand side of the constraints in (7).

To determine $P^\dagger$ we now need to find all solutions of $\mathcal{P}(x)$ which are in $[0, 1]$. After re-substituting for the $x_j = P^\dagger(\omega_j)$, exactly one of these solutions gives rise to x_j which are all in the

unit interval $[0,1]$ and which solve (20), because the entropy function is convex on line-segments in $\mathbb{P}$ and there hence exists a unique maximum in the interior of $\mathbb{P} \cap \mathcal{L}$. ■

Appendix D. Proof of Proposition 25

Proposition 25. *Under the assumptions of Theorem 24 and the assumption that $N = 3$ an OBN can be obtained by simple algebraic means without solving an optimisation problem.*

Proof: Enumerate the eight assignments $v@\{A_0, A_1, A_2\}$ as follows: assignment v_1 is $a_0a_1a_1$, v_2 is $a_0a_1\bar{a}_2$, and so on. Since $P^\dagger$ has to match the marginal the distributions P_i^*, where they are defined, $P^\dagger$ has to satisfy the following 12 linear constraints and no further constraints:

$$\text{constraints from } DS_1 \text{ on } P^\dagger \begin{cases} P^\dagger(v_1) + P^\dagger(v_2) = P_1^*(a_0a_1) =: a \\ P^\dagger(v_3) + P^\dagger(v_4) = P_1^*(a_0\bar{a}_1) =: b \\ P^\dagger(v_5) + P^\dagger(v_6) = P_1^*(\bar{a}_0a_1) =: c \\ P^\dagger(v_7) + P^\dagger(v_8) = P_1^*(\bar{a}_0\bar{a}_1) =: d \end{cases}$$

$$\text{constraints from } DS_2 \text{ on } P^\dagger \begin{cases} P^\dagger(v_1) + P^\dagger(v_3) = P_2^*(a_0a_2) = a + b - f \\ P^\dagger(v_2) + P^\dagger(v_4) = P_2^*(a_0\bar{a}_2) =: f \\ P^\dagger(v_5) + P^\dagger(v_7) = P_2^*(\bar{a}_0a_2) = c + d - e \\ P^\dagger(v_6) + P^\dagger(v_8) = P_2^*(\bar{a}_0\bar{a}_2) =: e \end{cases}$$

$$\text{constraints from } DS_3 \text{ on } P^\dagger \begin{cases} P^\dagger(v_1) + P^\dagger(v_5) = P_3^*(a_1a_2) = a + c - e - f + g \\ P^\dagger(v_2) + P^\dagger(v_6) = P_3^*(a_1\bar{a}_2) = e + f - g \\ P^\dagger(v_3) + P^\dagger(v_7) = P_3^*(\bar{a}_1a_2) = b + d - g \\ P^\dagger(v_4) + P^\dagger(v_8) = P_3^*(\bar{a}_1\bar{a}_2) =: g \ . \end{cases}$$

The constraint that $P^\dagger$ is a probability function, $\sum_{i=1}^{8} P^\dagger(v_i) = 1$, is entailed by the above constraints, since $a + b + c + d = 1$.

Following the proof of Theorem 24 we first obtain the system of logically equivalent system of seven inhomogeneous equations for eight variables

$$\begin{aligned} x_1 - x_4 &= a - f & \qquad x_5 - x_8 &= c - e \\ x_2 + x_4 &= f & \qquad x_6 + x_8 &= e \\ x_3 + x_4 &= b & \qquad x_7 + x_8 &= d \\ x_4 + x_8 &= g \ , \end{aligned}$$

which we transform to the following system of equations

$$\begin{aligned} x_1 + x_8 &= a - f + g & \qquad x_4 + x_8 &= g \\ x_2 - x_8 &= f - g & \qquad x_5 - x_8 &= c - e \\ x_3 - x_8 &= b - g & \qquad x_6 + x_8 &= e \\ x_7 + x_8 &= d \ . \end{aligned} \tag{23}$$

Letting $x := x_8$ we can express the entropy of a probability function P solving this system of equations as

$$\begin{aligned} H(x) = &-(a-f+g-x)\log(a-f+g-x) - (f-g+x)\log(f-g+x) \\ &- (b-g-x)\log(b-g+x) - (g-x)\log(g-x) - (c-e+x)\log(c-e+x) \\ &- (e-x)\log(e-x) - (d-x)\log(d-x) - x\log(x) \ . \end{aligned}$$

Equating the derivative of $H(x)$ with respect to x to 0 we obtain

$$\begin{aligned} 0 = &\log(a-f+g-x) - \log(f-g+x) - \log(b-g+x) + \log(g-x) \\ &- \log(c-e+x) + \log(e-x) + \log(d-x) - \log(x) \ , \end{aligned}$$

which holds, if and only if

$$1 = \frac{-a+f-g+x}{f-g+x} \cdot \frac{-g+x}{b-g+x} \cdot \frac{-e+x}{c-e+x} \cdot \frac{-d+x}{x} \ .$$

To ease the notation we let $\varphi_1 := -a+f-g$, $\varphi_2 := -g$ and $\varphi_3 := -e$ and obtain

$$1 = \frac{\varphi_1+x}{a+\varphi_1+x} \cdot \frac{\varphi_2+x}{b+\varphi_2+x} \cdot \frac{\varphi_3+x}{c+\varphi_3+x} \cdot \frac{-d+x}{x} \ .$$

We obtain the polynomial $\mathcal{P}(x)$ as

$$\mathcal{P}(x) = x^3 + \beta x^2 + \gamma x + \delta = 0 \tag{24}$$

with

$$\begin{aligned} \beta :=& ab + ac + bc + \varphi_1 + \varphi_2 + \varphi_3 - \varphi_1 a - \varphi_2 b - \varphi_3 c \\ \gamma :=& abc + \varphi_1 bc + \varphi_2 ac + \varphi_3 ab + (1-a-b)\varphi_1\varphi_2 + (1-a-c)\varphi_1\varphi_3 + (1-b-c)\varphi_2\varphi_3 \\ \delta :=& \varphi_1\varphi_2\varphi_3 d \ . \end{aligned}$$

If x_* is a double root of $\mathcal{P}(x)$, then $H(x_*)$ is a saddle-point. We are looking for the unique maximum of $H(x)$ and can hence ignore double roots of $\mathcal{P}(x)$. The solutions of the cubic $\mathcal{P}(x)$ which are not saddle-points can be found in many textbooks, for example in (Press et al., 2007, p. 228).

The discriminant Δ of the polynomial $\mathcal{P}(x)$ and the auxiliary values p, q are defined as follows

$$\begin{aligned} \Delta :=& \frac{27\delta^2 + 4\beta^3\delta - 18\beta\gamma\delta + 4\gamma^3 - \beta^2\gamma^2}{108} \\ p :=& \frac{9\gamma - 3\beta^2}{9} = \frac{3\gamma - \beta^2}{3} \\ q :=& \frac{2\beta^3 - 9\beta\gamma + 27\delta}{27} \\ \Delta =& (q/2)^2 + (p/3)^3 \ . \end{aligned}$$

If $\underline{\Delta > 0}$, then

$$x^* = \sqrt[3]{-\frac{q}{2} + \sqrt{\Delta}} + \sqrt[3]{-\frac{q}{2} - \sqrt{\Delta}} - \frac{\beta}{3} \ ,$$

where the third roots must be the real roots.

If $\underline{\Delta = 0}$, then the roots of $\mathcal{P}(x)$ are

$$x^* = \begin{cases} -\frac{\beta}{3} & \text{if } \Delta = p = q = 0 \\ \frac{\beta^3 - 4\beta\gamma + 9\delta}{3\gamma - \beta^2} & \text{if } \Delta = 0 \text{ and } p^2 + q^2 > 0 \end{cases} .$$

If $\underline{\Delta < 0}$, then x^* is one of the following three values

$$x^* = \begin{cases} - \sqrt{-\frac{4p}{3}} \cdot \cos\Big(\frac{\pi}{3} + \frac{1}{3}\arccos\Big(-\frac{q}{2} \cdot \sqrt{-\frac{27}{p^3}}\Big)\Big) - \frac{\beta}{3} \\ \sqrt{-\frac{4p}{3}} \cdot \cos\Big(\frac{1}{3}\arccos\Big(-\frac{q}{2} \cdot \sqrt{-\frac{27}{p^3}}\Big)\Big) - \frac{\beta}{3} \\ - \sqrt{-\frac{4p}{3}} \cdot \cos\Big(-\frac{\pi}{3} + \frac{1}{3}\arccos\Big(-\frac{q}{2} \cdot \sqrt{-\frac{27}{p^3}}\Big)\Big) - \frac{\beta}{3} \end{cases} .$$

$P^\dagger$ can now quickly be computed. Simply check which of the six possible values for x^* is a root of $\mathcal{P}(x)$ and gives rise to a probability function, i.e., $0 \leq x_i \leq 1$ for all $1 \leq i \leq 8$. In particular, we find $P^\dagger$ without solving an optimisation problem. ■

Appendix E. Matlab Code for the Examples

We here give the Matlab code for our examples.
First we provide the code for the m.file:

```
function [x,fval,exitflag,output,lambda,grad,hessian] =
 maxent(N,Aineqinput,bineqinput,Aeqinput,beqinput)
% Returns [P m] where P achives maximum entropy m
% Input dimentions of the probabity function
% and optional inequality constraints (Aineq,bineq)
% and optional equality constraints (Aeq,beq)
tic
if nargin<=4
    Aeqinput=[];
    beqinput=[];
end
if nargin<=2
    Aineqinput=[];
    bineqinput=[];
end
x0 = ones(1,N)/N;
Ai = eye(N).*-1;
bi = zeros(1,N);
Aineq = [Ai; Aineqinput];
bineq = [bi bineqinput];
Ae = ones(1,N);
be = [1];
Aeq = [Ae; Aeqinput];
beq = [be beqinput];
lb = zeros(1,N);
```

```
options = optimset;
options = optimset(options,'Display', 'off');
options = optimset(options,'FunValCheck', 'on');
options = optimset(options,'Algorithm', 'interior-point', 'MaxIter', 1000);
[x,fval,exitflag,output,lambda,grad,hessian] = ...
fmincon(@NegEntropy,x0,Aineq,bineq,Aeq,beq,lb,[],[],options);
toc
```

Next we present the code to run Example 26.
First the brute-force computation of $P^{\dagger}$:

```
clear variables
format long
maxenttictoc([8],[],[],...
[...
1 1 0 0 0 0 0 0;...
0 0 1 1 0 0 0 0;...
0 0 0 0 1 1 0 0;...
0 0 0 0 0 0 1 1;...
1 0 1 0 0 0 0 0;...
0 1 0 1 0 0 0 0;...
0 0 0 0 1 0 1 0;...
0 0 0 0 0 1 0 1;...
1 0 0 0 1 0 0 0;...
0 1 0 0 0 1 0 0;...
0 0 1 0 0 0 1 0;...
0 0 0 1 0 0 0 1;...
],...
[55/100 10/100 33/100 2/100 64/100 1/100 7/100 28/100
61/100 27/100 10/100 2/100])

Pˆdagger=0.542471272915907   0.007528727084093   0.097528727084093
         0.002471272915907   0.067528727084093   0.262471272915907
         0.002471272915907   0.017528727084093
```

The polynomial $\mathcal{P}(x)$, the auxiliary parameters and the roots:

```
format long
clear variables
a=55/100; b=10/100; c=33/100; d=2/100; e=28/100; f=1/100; g=2/100;
phi_1=-a+f-g;
phi_2=-g;
phi_3=-e;
beta=a * b+a * c+b * c+phi_1+phi_2+phi_3-phi_1 * a-phi_2 * b-phi_3 * c;
gamma=a * b * c+phi_1 * b * c+phi_2 * a * c+phi_3 * a * b+(1-a-b) ...
* phi_1 * phi_2+(1-a-c) * phi_1 * phi_3...
+(1-b-c) * phi_2 * phi_3;
delta=phi_1*phi_2*phi_3 * d;
Discreminant=(27 * deltaˆ2+4 * betaˆ3 * delta-18 * beta * gamma ...
* delta+4 * gammaˆ3-betaˆ2 * gammaˆ2)/108
p=(9 * gamma-3 * betaˆ2)/9
q=(1/27) * ((2 * betaˆ3)-(9 * beta * gamma)+(27 * delta))
```

```
-sqrt(-(4*p)/3)*cos((1/3)*acos(-(q/2)*sqrt(-27/p^3))+(pi/3))-beta/3
-sqrt(-(4*p)/3)*cos((1/3)*acos(-(q/2)*sqrt(-27/p^3))-(pi/3))-beta/3
sqrt(-(4*p)/3)*cos((1/3)*acos(-(q/2)*sqrt(-27/p^3)))-beta/3

Discreminant=-1.097210366162764e-10
p =  -0.005225870000000
q =  -1.438901660000006e-04
ans =0.024494218202939
ans =0.017529189605105
ans =0.146076592191956
```

Computations of the entropy are executed as follows:

```
 0.542471272915907.*log(0.542471272915907)...
+0.007528727084093.*log(0.007528727084093)...
+0.097528727084093.*log(0.097528727084093)...
+0.002471272915907.*log(0.002471272915907)...
+0.067528727084093.*log(0.067528727084093)...
+0.262471272915907.*log(0.262471272915907)...
+0.002471272915907.*log(0.002471272915907)...
+0.017528727084093.*log(0.017528727084093)

x=0.017528727084093;
a=55/100; b=10/100; c=33/100; d=2/100; e=28/100; f=1/100; g=2/100;
(a-f+g-x).*log(a-f+g-x)+(f-g+x).*log(f-g+x)+(b-g+x).*log(b-g+x)...
+(g-x).*log(g-x)+(c-e+x).*log(c-e+x)+(e-x).*log(e-x)+(d-x)...
.*log(d-x)+x.*log(x)
```

Example 27.

Here is the code for the brute-force computation of $P^{\dagger}$:

```
clear variables
format long
maxenttictoc([8],[],[],...
[...
1 1 0 0 0 0 0 0;...
0 0 1 1 0 0 0 0;...
0 0 0 0 1 1 0 0;...
0 0 0 0 0 0 1 1;...
1 0 1 0 0 0 0 0;...
0 1 0 1 0 0 0 0;...
0 0 0 0 1 0 1 0;...
0 0 0 0 0 1 0 1;...
1 0 0 0 1 0 0 0;...
0 1 0 0 0 1 0 0;...
0 0 1 0 0 0 1 0;...
0 0 0 1 0 0 0 1;...
],...
[1/13 3/13 5/13 4/13 1/13 3/13 2/13 7/13 2/13 4/13 1/13 6/13])

P^dagger=0.034394277588952   0.042528799334125   0.042528799334124
         0.188240431435106   0.119451876257201   0.265163508358183
         0.034394277588952   0.273298030103355
```

The polynomial $\mathcal{P}(x)$, the auxiliary parameters and the roots:

```
tic
format long
clear variables
a=1/13; b=3/13; c=5/13; d=4/13; e=7/13; f=3/13; g=6/13;
phi_1=-a+f-g;
phi_2=-g;
phi_3=-e;
beta=a * b+a * c+b * c+phi_1+phi_2+phi_3-phi_1 * a-phi_2 * b-phi_3 * c;
gamma=a * b * c+phi_1 * b * c+phi_2 * a * c+phi_3 * a * b+(1-a-b) ...
* phi_1 * phi_2+(1-a-c) * phi_1 * phi_3...
+(1-b-c) * phi_2 * phi_3;
delta=phi_1*phi_2*phi_3 * d;
Discreminant=(27 * delta^2+4 * beta^3 * delta-18 * beta * gamma ...
* delta+4 * gamma^3-beta^2 * gamma^2)/108
p=(9 * gamma-3 * beta^2)/9
q=(1/27) * ((2 * beta^3)-(9 * beta * gamma)+(27 * delta))
root=(-q/2+(Discreminant)^(1/2))^(1/3)...
-(+q/2+(Discreminant)^(1/2))^(1/3)-(beta/3)
toc

Discreminant =1.525101176783421e-08
p =   0.007387696509226
q =   3.563430829767589e-05
root =0.273298090801853
```

Computations of the entropy are executed as follows:

```
 0.034394277588952.*log(0.034394277588952)...
+0.042528799334125.*log(0.042528799334125)...
+0.042528799334124.*log(0.042528799334124)...
+0.188240431435106.*log(0.188240431435106)...
+0.119451876257201.*log(0.119451876257201)...
+0.265163508358183.*log(0.265163508358183)...
+0.034394277588952.*log(0.034394277588952)...
+0.273298030103355.*log(0.273298030103355)

x=0.273298090801853;
a=1/13; b=3/13; c=5/13; d=4/13; e=7/13; f=3/13; g=6/13;
(a-f+g-x).*log(a-f+g-x)+(f-g+x).*log(f-g+x)+(b-g+x).*log(b-g+x)...
+(g-x).*log(g-x)+(c-e+x).*log(c-e+x)+(e-x).*log(e-x)...
+(d-x).*log(d-x)+x.*log(x)
```

References

Abramov, R. V. (2010). The multidimensional maximum entropy moment problem: a review on numerical methods. *Communications in Mathematical Sciences*, 8(2):377–392.

Adamčík, M. (2016). On the Applicability of the 'Number of Possible States' Argument in Multi-Expert Reasoning. *Journal of Applied Logic*, 19, Part 1:20–49.

Aliferis, C. F., Statnikov, A., Tsamardinos, I., Mani, S., and Koutsoukos, X. D. (2010). Local Causal and Markov Blanket Induction for Causal Discovery and Feature Selection for Classification Part I: Algorithms and Empirical Evaluation. *Journal of Machine Learning Research*, 11(7):171–234.

Aliferis, C. F., Tsamardinos, I., Statnikov, A., and Brown, L. E. (2003). Causal explorer: a causal probabilistic network learning toolkit for biomedical discovery. In Valafar, F., editor, *Proceedings of the 2003 International Conference on Mathematics and Engineering Techniques in Medicine and Biological Sciences (METMBS)*, Athens, Georgia. CSREA Press.

Bai, J. P. and Abernethy, D. R. (2013). Systems pharmacology to predict drug toxicity: integration across levels of biological organization. *Annual review of pharmacology and toxicology*, 53:451–473.

Balasubramanian, V. (2005). MDL, Bayesian Inference, and the Geometry of the Space of Probability Distributions. In Grüwald, P. D., Myung, I. J., and Pitt, M. A., editors, *Advances in Minimum Description Length*, chapter 3, pages 81–98. MIT Press.

Balestrino, A., Caiti, A., and Crisostomi, E. (2006). Efficient numerical approximation of maximum entropy estimates. *International Journal of Control*, 79(9):1145–1155.

Bareinboim, E. and Pearl, J. (2016). Causal inference and the data-fusion problem. *Proceedings of the National Academy of Sciences*, 113(27):7345–7352.

Berger, A., Della Pietra, S., and Della Pietra, V. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Berry, A. (1999). A wide-range efficient algorithm for minimal triangulation. In *Proceedings of SODA*.

Berry, A., Blair, J. R. S., Heggernes, P., and Peyton, B. W. (2004). Maximum cardinality search for computing minimal triangulations of graphs. *Algorithmica*, 39(4):287–298.

Berry, A., Bordat, J.-P., Heggernes, P., Simonet, G., and Villanger, Y. (2006). A wide-range algorithm for minimal triangulation from an arbitrary ordering. *Journal of Algorithms*, 58(1):33–66.

Berry, A. and Pogorelcnik, R. (2011). A simple algorithm to generate the minimal separators and the maximal cliques of a chordal graph. *Information Processing Letters*, 111(11):508–511.

Borenstein, M., Hedges, L., Higgins, J., and Hannah, R. (2009). *Introduction to Meta-Analysis*. Wiley, Chichester.

Bromberg, F., Margaritis, D., and Honavar, V. (2009). Efficient Markov Network Structure Discovery Using Independence Tests. *Journal of Artificial Intelligence Research*, 35:449–484.

Caticha, A. (2013). Towards an Informational Pragmatic Realism. *Minds and Machines*, 24(1):37–70.

Caticha, A. (2014). Entropic Dynamics: an inference approach to quantum theory, time and measurement. *Journal of Physics: Conference Series*, 504(1):012009.

Cheeseman, P. (1983). A method of computing generalised Bayesian probability values for expert systems. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 198–202. William Kaufmann, Los Altos CA.

Chen, B., Hu, J., and Zhu, Y. (2010). Computing maximum entropy densities: A hybrid approach. *Signal Processing: An International Journal*, 4(2):114–122.

Danks, D. (2002). Learning the Causal Structure of Overlapping Variable Sets. In Lange, S., Satoh, K., and Smith, C. H., editors, *Proceedings of DS*, pages 178–191. Springer.

Danks, D., Glymour, C., and Tillman, R. E. (2008). Integrating Locally Learned Causal Structures with Overlapping Variables. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Proceedings of NIPS*, pages 1665–1672. Curran Associates.

De Pretis, F., Landes, J., and Peden, W. J. (2021). Artificial Intelligence Methods For a Bayesian Epistemology-Powered Evidence Evaluation. *Journal of Evaluation in Clinical Practice*, 27(3):504–512.

Dor, D. and Tarsi, M. (1992). A simple algorithm to construct a consistent extension of a partially oriented graph. Technical report, UCLA.

D'Orazio, M., Di Zio, M., and Scanu, M. (2006). *Statistical Matching: Theory and Practice*. Wiley.

Endres, E. and Augustin, T. (2016). Statistical Matching of Discrete Data by Bayesian Networks. In Antonucci, A., Corani, G., and de Campos, C. P., editors, *Proceedings of PGM*, volume 52, pages 159–170.

Fulkerson, D. R. and Gross, O. (1965). Incidence matrices and interval graphs. *Pacific Journal of Mathematics*, 15(3):835–855.

Gámez, J. A., Mateo, J. L., and Puerta, J. M. (2010). Learning Bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 22(1-2):106–148.

Goldman, S. A. (1987). Efficient methods for calculating maximum entropy distributions. Master's thesis, Electrical Engineering and Computer Science, Massachusetts Institute of Technology.

Goldman, S. A. and Rivest, R. (1988). A non-iterative maximum entropy algorithm. In Kanal, L. and Lemmer, J., editors, *Uncertainty in Artificial Intelligence 2*, pages 133–148. Elsevier, North-Holland.

Grove, A. J., Halpern, J. Y., and Koller, D. (1994). Random Worlds and Maximum Entropy. *Journal of Artificial Intelligence Research*, 2(1):33–88.

Halpern, J. Y. and Koller, D. (2004). Representation Dependence in Probabilistic Inference. *Journal of Artificial Intelligence Research*, 21:319–356.

Hauser, A. and Bühlmann, P. (2014). Jointly interventional and observational data: estimation of interventional markov equivalence classes of directed acyclic graphs. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 77(1):291–318.

Heggernes, P. (2006). Minimal triangulations of graphs: A survey. *Discrete Mathematics*, 306(3):297–317.

Heinze-Deml, C., Maathuis, M. H., and Meinshausen, N. (2018). Causal Structure Learning. *Annual Review of Statistics and Its Application*, 5(1):371–391.

Jaynes, E. T. (1957). Information theory and statistical mechanics. *The Physical Review*, 106(4):620–630.

Jaynes, E. T. (2003). *Probability theory: the logic of science*. Cambridge University Press, Cambridge.

Jenkins, M. A. and Traub, J. F. (1970). A Three-Stage Algorithm for Real Polynomials Using Quadratic Iteration. *SIAM Journal on Numerical Analysis*, 7(4):545–566.

Kern-Isberner, G. (1998). Characterizing the principle of minimum cross-entropy within a conditional-logical framework. *Artificial Intelligence*, 98(1-2):169–208.

Kern-Isberner, G. and Lukasiewicz, T. (2004). Combining probabilistic logic programming with the power of maximum entropy. *Artificial Intelligence*, 157(1-2):139–202.

Kern-Isberner, G. and Rödder, W. (2004). Belief revision and information fusion on optimum entropy. *International Journal of Intelligent Systems*, 19(9):837–857.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models*. MIT Press, Cambridge, MA.

Landes, J. and Williamson, J. (2013). Objective Bayesianism and the maximum entropy principle. *Entropy*, 15(9):3528–3591.

Landes, J. and Williamson, J. (2015). Justifying objective Bayesianism on predicate languages. *Entropy*, 17(4):2459–2543.

Landes, J. and Williamson, J. (2016). Objective Bayesian nets from consistent datasets. In Giffin, A. and Knuth, K. H., editors, *Proceedings of MaxEnt*, volume 1757, pages 020007–1 – 020007–8. AIP.

Lauritzen, S. L. (1996). *Graphical models*. Clarendon Press, Oxford.

Little, R. J. and Rubin, D. A. (2014). *Statistical Analysis with Missing Data*. Wiley, Hoboken, 2 edition.

Liu, Z., Malone, B., and Yuan, C. (2012). Empirical evaluation of scoring functions for Bayesian network model selection. *BMC Bioinformatics*, 13(S15).

Lukasiewicz, T. (2000). Credal Networks under Maximum Entropy. In *Proceedings of UAI*, pages 363–370, San Francisco. Morgan Kaufmann.

Malone, B., Järvisalo, M., and Myllymäki, P. (2015). Impact of Learning Strategies on the Quality of Bayesian Networks: An Empirical Evaluation. In *Proceedings UAI*, pages 562–571, Arlington, Virginia, USA. AUAI Press.

Maung, I. and Paris, J. B. (1990). A note on the infeasibility of some inference processes. *International Journal of Intelligent Systems*, 5(5):595–603.

Mayo-Wilson, C. (2019). Causal identifiability and piecemeal experimentation. *Synthese*, 196:3029–3065.

Nagl, S., Williams, M., and Williamson, J. (2008). Objective Bayesian Nets for Systems Modelling and Prognosis in Breast Cancer. In Holmes, D. and Jain, L., editors, *Innovations in Bayesian Networks*, volume 156 of *Studies in Computational Intelligence*, pages 131–167. Springer.

Nandy, P., Hauser, A., and Maathuis, M. H. (2018). High-dimensional consistency in score-based and hybrid structure learning. *Annals of Statistics*, 46(6A):3151–3183.

Neapolitan, R. E. (1990). *Probabilistic reasoning in expert systems: theory and algorithms*. Wiley, New York.

Neapolitan, R. E. (2003). *Learning Bayesian Networks*. Pearson, Upper Saddle River.

Nielsen, R. A. (2016). Case Selection via Matching. *Sociological Methods & Research*, 45(3):569–597.

Ordyniak, S. and Szeider, S. (2013). Parameterized Complexity Results for Exact Bayesian Network Structure Learning. *Journal of Artificial Intelligence Research*, 46:263–302.

Ormoneit, D. and White, H. (1999). An efficient algorithm to compute maximum entropy densities. *Econometric Reviews*, 18(2):127–140.

Paris, J. B. (1994). *The Uncertain Reasoner's Companion*. Cambridge University Press, Cambridge.

Paris, J. B. (2005). On filling-in missing conditional probabilities in causal networks. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 13(3):263–280.

Paris, J. B. (2014). What you see is what you get. *Entropy*, 16(11):6186–6194.

Paris, J. B. and Vencovská, A. (1990). A note on the inevitability of maximum entropy. *International Journal of Approximate Reasoning*, 4(3):183–223.

Paris, J. B. and Vencovská, A. (1997). In defense of the maximum entropy inference process. *International Journal of Approximate Reasoning*, 17(1):77–103.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, San Mateo CA.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (2007). *Numerical Recipes*. Cambridge University Press, Cambridge.

Ras, G., Xie, N., Gerven, M. V., and Doran, D. (2022). Explainable Deep Learning: A Field Guide for the Uninitiated. *Journal of Artificial Intelligence Research*, 73:329–397.

Raskutti, G. and Uhler, C. (2018). Learning directed acyclic graph models based on sparsest permutations. *Stat*, 7(1):e183.

Rosenkrantz, R. D. (1977). *Inference, method and decision: towards a Bayesian philosophy of science*. Reidel, Dordrecht.

Schleicher, J., Eklund, J., Barnes, M. D., Geldmann, J., Oldekop, J. A., and Jones, J. P. G. (2020). Statistical matching for conservation science. *Conservation Biology*, 34(3):538–549.

Scutari, M. (2010). Learning bayesian networks with the bnlearn R package. *Journal of Statistical Software*, 35(i03).

Scutari, M., Graafland, C. E., and Gutiérrez, J. M. (2019). Who learns better Bayesian network structures: Accuracy and speed of structure learning algorithms. *International Journal of Approximate Reasoning*, 115:235–253.

Srebro, N. (2003). Maximum likelihood bounded tree-width markov networks. *Artificial Intelligence*, 143(1):123 – 138.

Thimm, M., Finthammer, M., Loh, S., Kern-Isberner, G., and Beierle, C. (2010). A System for Relational Probabilistic Reasoning on Maximum Entropy. In Guesgen, H. W. and Murray, R. C., editors, *Proceedings FLAIRS*. AAAI Press.

Tillman, R. and Spirtes, P. (2011). Learning equivalence classes of acyclic models with latent and selection variables from multiple datasets with overlapping variables. *JMLR Workshop and Conference Proceedings*, 15.

Tillman, R. E. and Eberhardt, F. (2014). Learning causal structure from multiple datasets with similar variable sets. *Behaviormetrika*, 41(1):41–64.

Triantafillou, S. and Tsamardinos, I. (2015). Constraint-based Causal Discovery from Multiple Interventions over Overlapping Variable Sets. *Journal of Machine Learning Research*, 16:2147–2205.

Triantafillou, S., Tsamardinos, I., and Tollis, I. (2010). Learning Causal Structure from Overlapping Variable Sets. *Journal of Machine Learning Research*, 9:860–867.

Tribus, M. (1969). *Rational descriptions, decisions and designs*. Pergamon Press, New York.

Tricco, A. C., Antony, J., Soobiah, C., Kastner, M., MacDonald, H., Cogo, E., Lillie, E., Tran, J., and Straus, S. E. (2016). Knowledge synthesis methods for integrating qualitative and quantitative data: a scoping review reveals poor operationalization of the methodological steps. *Journal of Clinical Epidemiology*, 73:29–35.

Tsamardinos, I., Brown, L., and Aliferis, C. (2006). The max-min hill-climbing Bayesian network structure learning algorithm. *Machine Learning*, 65(1):31–78.

Tsamardinos, I., Triantafillou, S., and Lagani, V. (2012). Towards Integrative Causal Analysis of Heterogeneous Data Sets and Studies. *Journal of Machine Learning Research*, 13:1097–1157.

Vantaggi, B. (2008). Statistical matching of multiple sources: A look through coherence. *International Journal of Approximate Reasoning*, 49(3):701–711.

Williamson, J. (2002). Maximising entropy efficiently. *Electronic Transactions in Artificial Intelligence Journal*, 6. www.etaij.org.

Williamson, J. (2005a). *Bayesian Nets and Causality*. Oxford University Press, Oxford.

Williamson, J. (2005b). Objective Bayesian nets. In Artemov, S., Barringer, H., d'Avila Garcez, A. S., Lamb, L. C., and Woods, J., editors, *We Will Show Them! Essays in Honour of Dov Gabbay*, volume 2, pages 713–730. College Publications, London.

Williamson, J. (2010). *In defence of objective Bayesianism*. Oxford University Press, Oxford.

Williamson, J. (2017). Models in Systems Medicine. *Disputatio*, 9(47):429–469.

Xie, X. and Geng, Z. (2008). A Recursive Method for Structural Learning of Directed Acyclic Graphs. *Journal of Machine Learning Research*, 9(14):459–483.